OBSTACLE DETECTION USING A MONOCULAR CAMERA

A Thesis
Presented to
The Academic Faculty

by

Rostislav Goroshin

In Partial Fulfillment
Of the Requirements for the Degree
Master of Science in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

August, 2008

OBSTACLE DETECTION USING A MONOCULAR CAMERA

Approved by:

Dr. P.A. Vela
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. T. Collins
Georgia Tech Research Institute
Georgia Institute of Technology

Dr. Ayanna Howard
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: May 14, 2008

ACKNOWLEDGEMENTS

I thank my research advisor, Dr. P.A. Vela for his guidance, support, and instilling in me the belief that careful modeling and persistence is key. I thank Dr. Collins for many useful comments during the writing process. I would also like to thank William Mantzel for many useful conversations regarding computer vision, and his friendship. I am grateful to my lab mates, Omar and Ibrahima, for their advice and many interesting conversations and group-meetings. Finally, I thank my parents for their unwaivering support and encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

SUMMARY

The objective of this thesis is to develop a general obstacle segmentation algorithm for use on board a ground based unmanned vehicle (GUV). The algorithm processes video data captured by a single monocular camera mounted on the GUV. We make the assumption that the GUV moves on a locally planar surface, representing the ground plane. We start by deriving the equations of the expected motion field (observed by the camera) induced by the motion of the robot on the ground plane. Given an initial view of a presumably static scene, this motion field is used to generate a predicted view of the same scene after a known camera displacement. This predicted image is compared to the actual image taken at the new camera location by means of an optical flow calculation. Because the planar assumption is used to generate the predicted image, portions of the image which mismatch the prediction correspond to salient feature points on objects which lie above or below the ground plane, we consider these objects obstacles for the GUV. We assume that these salient feature points (called “seed pixels”) capture the color statistics of the obstacle and use them to initialize a Bayesian region growing routine to generate a full obstacle segmentation. Alignment of the seed pixels with the obstacle is not guaranteed due to the aperture problem, however successful segmentations were obtained for natural scenes. The algorithm was tested off line using video captured by a camera mounted on a GUV.

CHAPTER 1

INTRODUCTION

The historic trend in the mobile robotics community has been to equip the mobile agent with a number of specialized and often expensive sensors for navigation. Examples include LIDAR, Radar, GPS, and Stereo Cameras[9]. The main objective of this work is to investigate the limitations of a single monocular camera acting as the only navigation sensor for a ground based unmanned vehicle (GUV). Specifically, we focus on the task of obstacle detection. In the context of ground vehicles, obstacles are defined as objects which protrude from the ground plane. The proposed approach is inspired by the fact that a human operator can remotely navigate a ground vehicle (albeit, sometimes with moderate success) in an unknown environment using only monocular vision as feedback. There are several advantages for using a relatively simple, passive sensor like a monocular camera, reduction in cost and weight are two. Also in some covert or surveillance applications passive sensors are preferred.

Color video signals are rich in information about the surrounding world, however the burden of extracting relevant information for a particular application is passed on to higher level processing, usually performed at the software level. The extraction of relevant information about the surroundings from image or video signals is, by definition, associated with the discipline of computer vision[10]. In contrast to the paradigm in computer vision, traditional sensor design doctrine strives to design specialized sensors whose functionality is embedded in the physical properties of the sensor itself, and whose

signals require little or no post processing to extract the desired information. For example, a radar signal contains range information to a target object, the object's velocity information can be extracted through post processing, however the signal contains no information about the object's color. Conversely because an image is the two-dimensional projection of the three-dimensional world, it contains no depth (range) information (although some researchers have tried to extract it anyway[11]). However, depth information can be extracted from multiple images of the same scene taken from different viewpoints, this problem is called “structure from motion (SFM)”[2]. For the purpose of obstacle detection for a UGV the structure of the entire scene need not be known, it is sufficient to identify those objects which protrude from the ground plane enough to impede the vehicle's motion.

In this chapter the constraints used in the development of the proposed obstacle detection approach will be discussed. Next, previous work in the area of monocular obstacle detection will be reviewed. Lastly, an outline of the approach will be presented.

1.1 Underlying Constraints and Previous Work

Because the proposed obstacle detection algorithm is specifically tailored for ground-based vehicles we will assume that there exists a locally planar surface on which the vehicle is constrained to move. This is a commonly made assumption found in related literature, however it may be exploited in different ways. Ulrich and Nourbakhsh assume that the ground plane has a constant color distribution in their “Appearance-Based” obstacle detection scheme[12]. In the works of M. Grunewald[13] and Santos-Victor &

Sandini[18] the ground plane assumption is exploited in the geometric sense. The approach undertaken in this work will make use of both the appearance and geometric assumptions of the ground plane. Namely, obstacle objects differ in appearance from the ground plane *and* obstacle objects differ in geometry from the ground plane. We also make the "no overhanging obstacles" assumption [12], which assumes that all obstacles rest on the ground. The previous assumption is not necessary for obstacle detection but rather for calculating the distance to the obstacle which is necessary for navigation.

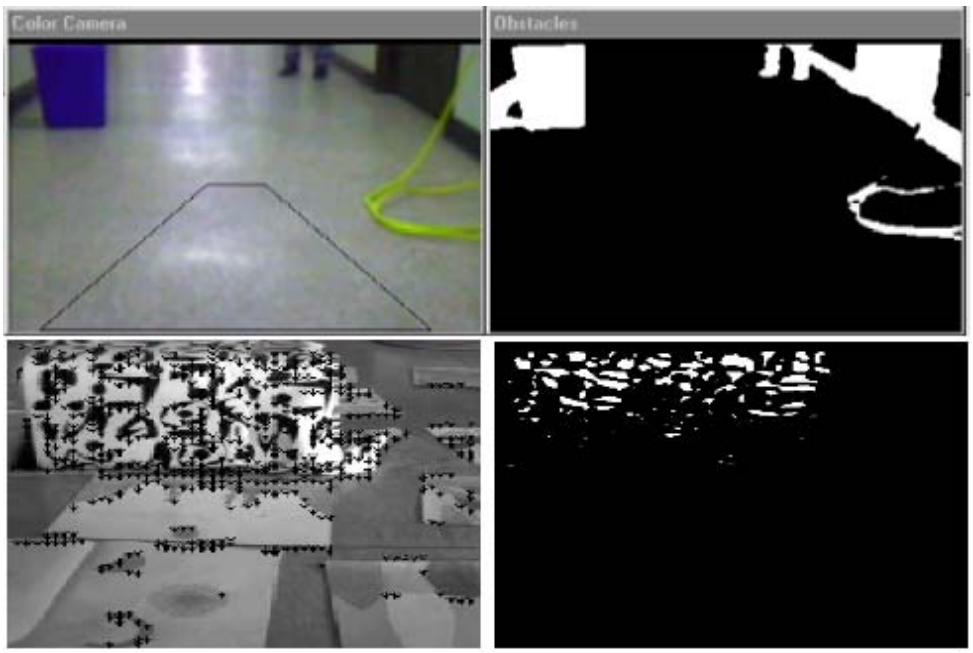


Fig. 1.1 Illustration of the difference between segmentation(a)[12] and detection(b)[14].

It should be mentioned that some works in this area attempt to achieve obstacle *detection* while others strive to obtain obstacle *segmentation*. Obstacle detection refers to obtaining a crude estimate of the obstacle's location in the image. Obstacle detection usually results in the detection of points on or near the obstacle. Segmentation attempts to identify the connected set of all points in the image corresponding to an obstacle. An example of obstacle segmentation (taken from [12]) is shown in Figure 1.1 (a) and an example of

obstacle detection (taken from [18]) is shown in Figure 1.1 (b). The proposed approach in this work is concerned with generating obstacle segmentations, therefore the terms "detection" and "segmentation" will be used interchangeably.

In their approach[12], Ulrich and Nourbakhsh assume that the ground plane has a constant and known color distribution. For example, if the sample of the ground distribution is taken to be the color histogram of the region enclosed by the trapezoid shown on the left of Figure 1.1 (a), then filtering the image to detect all objects whose color distributions mismatch the ground results in the segmentation shown in the right image of Figure 1.1 (a). The objects in white are considered to be obstacles. Clearly, this approach is limited by the knowledge of the color distribution of the ground terrain. That is, if the vehicle traverses terrain whose color distribution has not been stored a priori then the new ground terrain will appear to be an obstacle. Furthermore, if a flat, traversable region is encountered whose color distribution does not match the known ground distribution then it too will be segmented as an obstacle. This appearance based approach is not based on any geometric evidence that a given object is indeed an obstacle, however a full obstacle segmentation is achieved in some constrained cases. Conversely, the approach taken in [13] and [18] as well as in many other papers attempt to detect obstacles by their geometric properties rather than by their appearance based properties. One popular approach is to predict the perceived motion of the ground plane in the video sequence as the vehicle travels along the terrain. The motion in the video sequence is then measured by means of an optical flow calculation and compared to the predicted planar motion, those portions of the image which disagree with the predicted

planar motion must be non-planar and are labeled as obstacles. There are difficulties with measuring perceived motion via optical flow which will be discussed in Chapter 3. The result of such an approach is a sparse detection of the obstacle shown in Figure 1.1 (b).

1.2 Summary of the Approach

Approaches which achieve obstacle segmentation based on geometric evidence (such as [15]) use both motion and appearance information. This is because reliable motion estimates can only be obtained at salient feature points in the image, among other reasons which will be discussed in Chapter 3. Our approach will also use both motion and appearance information to generate full obstacle segmentations. The working assumptions used in the development of our approach are summarized:

1. The vehicle moves on a locally planar surface
2. The motion between neighboring frames of video is known
3. Obstacles differ in appearance (color) from the ground plane, however the appearance of the ground plane need not be known a-priori
4. There are no overhanging obstacles, that is, the obstacles rest on the ground plane

Our approach begins by making use of assumptions 1 and 2. Given an initial view of the scene we generate a predicted view of the same scene after some known camera displacement. The prediction assumes that the three-dimensional geometry of the scene is an infinite plane corresponding to the ground plane. The process of generating the predicted image is discussed in Chapter 2. The predicted image is compared to the actual image taken at the new camera location, this comparison reveals salient non-planar points

which are called “seed pixels”. This approach differs from the one taken in [13] and [18], in that the motion field between adjacent frames is not computed directly. The discussion on the utilization of motion information is found in Chapter 3. Next, under assumption 3, using the selected seed pixels (see Chapter 4) which we assume correspond to salient feature points of obstacles; we use Bayesian region growing to evolve a connected region which corresponds to the segmented obstacle. Segmentation is the subject of Chapter 4. Finally, we make use of assumption 4 to estimate the distance to the obstacle, this is discussed in Chapter 4.

CHAPTER 2

GENERATING THE PREDICTED IMAGE

The objective of the algorithm described in this chapter is to generate a predicted image of a presumably static scene after a known camera displacement. Given an initial image taken at some initial camera configuration (Xc_1, Yc_1, Zc_1) we wish to generate the predicted image of the same static scene taken at some new camera configuration (Xc_2, Yc_2, Zc_2), see Fig.2.1. These successive images may be regarded as frames in a video sequence recorded by a camera mounted on a mobile robot.

In order to generate the predicted image we require, for each pixel in the initial image, a two dimensional vector which determines the displacement of that pixel corresponding to the pixel's predicted motion between successive frames. The ensemble of these vectors will be called the “predicted motion field”. The problem of generating a predicted image of the scene is ill-posed unless an implicit assumption about the three-dimensional scene geometry is made. In this case we assume that the scene is a planar surface corresponding to the ground plane. The procedure for generating the predicted motion field consists of three main steps (Fig.2.4): (1) project the points on the image plane onto the ground plane using the perspective projection, (2) determine the coordinates of these projected points with respect to the new (displaced) camera coordinate frame, (3) project the points back onto the new image plane. In this manner a new image coordinate is assigned to every pixel in the original image, the difference between the new coordinate and the original coordinate determines the motion vector by which the corresponding pixel will be moved. Step (2) is detailed in Section 2.1, steps (2) & (3) are discussed in section 2.2.

The procedure is reiterated in Section 2.3 and depicted in Figure 2.4. After the predicted motion field is obtained the predicted image is generated by interpolation, as discussed in Section 2.3.

2.1 Describing Camera Motion

Let us use the homogeneous coordinate representation in order to facilitate the description of the rigid body transformations which we will use to describe the robot and camera motions. The homogeneous coordinate representation augments the vector representing the coordinate of a point by a unit component in the extra dimension. Therefore the homogeneous coordinate representation of a point in three dimensional spaces is **p** = (X, Y, Z, 1), similarly a vector is represented by **v** = (X, Y, Z, 0). Note that the difference of points results in a vector and the sum of a point and a vector results in another point[1].

Using this convention, the representation of a rigid transformation can be expressed as matrix multiplication. Suppose that the camera configuration is represented by a coordinate frame. In order to describe camera motion we must find the relationship between the initial and final coordinate frame of the camera, after some rotation and displacement.

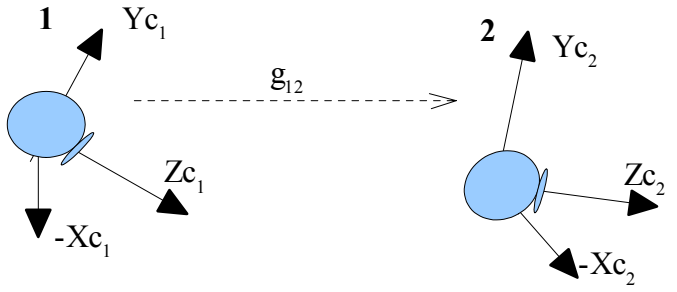


Fig. 2.1 Camera motion as described by a coordinate frame.

This relationship is described by a rigid body motion represented by a matrix (call it *g*). The *g*-matrix is itself composed of a rotation matrix (R), a vector describing translation (T), and is augmented for operation on points described in homogeneous coordinates. The structure of the *g*-matrix is given in equation (1) and its effect on an arbitrary point is given by Equation (2).

$$g=\begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}=\begin{bmatrix} r_{x1} & r_{x2} & r_{x3} & t_x \\ r_{y1} & r_{y2} & r_{y3} & t_y \\ r_{z1} & r_{z2} & r_{z3} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

$$g\,q=\begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}\begin{bmatrix} p \\ 1 \end{bmatrix}=\begin{bmatrix} Rp+T \\ 1 \end{bmatrix} \quad (2)$$

The rotation matrix-R describes a general rotation in three dimensions which can be decomposed into three separate rotations about three orthogonal axes, namely pitch(θ), yaw(φ) and roll(α) are rotations about the x,y,z axes respectively. These matrices are multiplied to obtain the resulting three dimensional rotational motion as shown in Equation 3 [1].

$$R=\begin{bmatrix} r_{x1} & r_{x2} & r_{x3} \\ r_{y1} & r_{y2} & r_{y3} \\ r_{z1} & r_{z2} & r_{z3} \end{bmatrix}=\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}\begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix}\begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3)$$

Similarly, successive rigid motions are described by the composition rule, that is if frame B is related to frame A by a rigid motion g_{ab} and frame C is related to frame B by rigid motion g_{bc} then frame C is related to frame A by the transformation $g_{ac}= g_{ab}\, g_{bc}$.

$$g_{ac}=g_{ab}\,g_{bc}=\begin{bmatrix} R_{ab}R_{bc} & R_{ab}T_{bc}+T_{ab} \\ 0 & 1 \end{bmatrix} \quad (4)$$

The last property of the *g*-matrix which we will use is the inversion property, namely if frame A is rigidly transformed to frame B by g_{ba} then frame B can be transformed to frame A by $g_{ab}=(g_{ba})^{-1}$. A short proof of this statement is given in equation (5), where we

use the fact that the rotation matrix is orthogonal, namely $R^{-1} = R^T$ [1].

$$g^{-1} = R^{-1}\begin{bmatrix} 1 & -T \\ 0 & R \end{bmatrix} = R^T\begin{bmatrix} 1 & -T \\ 0 & R \end{bmatrix} = \begin{bmatrix} R^T & -R^T T \\ 0 & 1 \end{bmatrix}$$

$$\textit{therefore}, \qquad (5)$$

$$(g^{-1})(gq) = \begin{bmatrix} R^T & -R^T T \\ 0 & 1 \end{bmatrix}\begin{bmatrix} Rp+T \\ 1 \end{bmatrix} = \begin{bmatrix} p \\ 1 \end{bmatrix}$$

Using the aforementioned properties we may now begin to describe the transformation which will allow us to generate the predicted image of a scene after a known camera displacement. The first step is to determine the motion of the camera. The camera motion is not directly measurable, but because the camera is rigidly mounted on the robot platform the motion of the camera is determined by the motion of the robot. Let g_{cr} describe the robot coordinate frame with respect to the camera coordinate frame. Also, let g_{r21} describe the final robot coordinate frame (after movement) with respect to the initial robot coordinate frame (before movement). The final coordinate frame of the camera (C_2) with respect to the initial coordinate frame of the camera (C_1) is therefore given by $g_{c12}=(g_{cr})^{-1}(g_{r21})(g_{cr})$, as shown in Fig. 2.2. Therefore a point **q** described with respect to C_1 would have coordinate $g_{c12}^{-1}\mathbf{q}$ with respect to C_2.

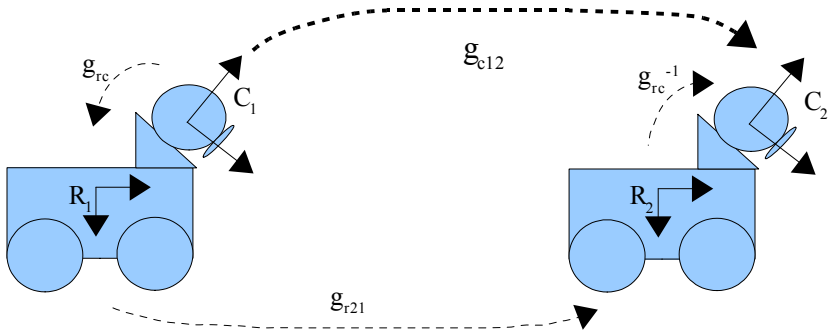


Fig. 2.2 Illustration of the relationship between the camera and robot motion.

$$g_{c12}^{-1} = [(g_{cr})^{-1}(g_{r12})(g_{cr})]^{-1} \qquad (6)$$

Note that the composition rule given by equation (4) implies that a pure rotation in one

reference frame may lead to a rotation *and* a translation in another reference frame rigidly attached to the first. Therefore for a given robot motion the corresponding camera motion is governed by g_{cr}. Assuming that the roll and yaw angles are zero, the relationship between robot (R) and camera (C) reference frame is illustrated in Fig.2.3. Note that dZ_R is the displacement from C to R along the Z_R axis, and dX_R is the displacement from C to R along the Y_R axis. X_C and X_R are parallel and point into the page. In order to describe g_{cr} however, dZ_R and dY_R are expressed in terms of Z_C and Y_C, as given by equation (7).

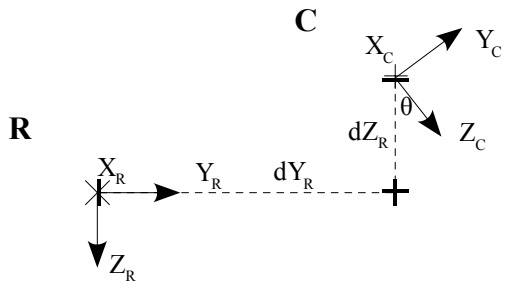


Fig. 2.3 Relationship between camera(C) and robot(R) coordinate frames.

$$\begin{bmatrix} t_y \\ t_x \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} -dZ_R \\ -dY_R \end{bmatrix} = \begin{bmatrix} -dY_R\cos\theta - dZ_R\sin\theta \\ -dY_R\sin\theta + dZ_R\cos\theta \end{bmatrix} \quad (7)$$

$$g_{cr}^{pitch} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & -dY_R\cos\theta - dZ_R\sin\theta \\ 0 & \sin\theta & \cos\theta & -dY_R\sin\theta + dZ_R\cos\theta \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (8)$$

The relationship between camera and robot reference frames illustrated in Fig.3 and given by equation (8) presuming that the two frames lie precisely in the same plane, that is, the yaw and roll angles between the frames are zero. This idealization is usually invalid, therefore the extra two degrees of freedom given by the yaw(φ) and roll(α) angles are taken into account by multiplying g_{cr} by the two more *g*-matrices repressing these rotations. The translational component of the transformation is purposely included in the last matrix (g_{cr}^{pitch}).

$$g_{cr} = g_{cr}^{yaw}\, g_{cr}^{roll}\, g_{cr}^{pitch} \quad (9)$$

The motion of the robot is described by g_{r12}. Assuming that robot moves on a planar surface, the displacement of the robot is described by a translation in the (X_R,Y_R)-plane and rotation in the direction defined by the Z_R-axis. It is convenient to describe the displacement of the robot by computing the *g*-matrix that relates the robot's current configuration to it's previous configuration. The current configuration is the configuration at which we wish to generate the predicted image based on the image captured in the previous configuration. Assuming that the robot has displaced by δx, δy meters along the X_R and Y_R axis respectively, and has rotated by an amount *r*-radians, then the g_{r12} -matrix is given by equation (10).

$$g_{r12} = \begin{bmatrix} \cos r & \sin r & 0 & \delta_x \\ -\sin r & \cos r & 0 & \delta_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (10)$$

Equations (9) and (10) are substituted into equation (6) to obtain the expression for the camera motion between the successive frames of video.

2.2 Perspective Projection

The transformation derived in the previous section (Equation 6) is applied to points in three dimensional space; therefore, every point on the image plane represented by two dimensional image coordinates (*u,v*) must be assigned three dimensional world coordinates (*X,Y,Z*). This is done by assuming that the image points are the perspective projection of a three dimensional planar surface. This implicit assumption allows the recovery of the three-dimensional world coordinates from the two-dimensional image coordinates.

The perspective camera model is described by Equation 11, which is based on the assumption that the world is observed by the camera from a single point behind the image plane called the focal point, and that all light rays pass through this point[2]. The constant *f* is the distance from the focal point to the image plane and is called the focal distance.

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} Xf/Z \\ Yf/Z \end{bmatrix} \quad (11)$$

Thus the world point with coordinate (*X,Y,Z*) described with respect to the camera coordinate frame will have image plane coordinate (*u,v*). The three-dimensional planar surface representing the ground plane is described by Equations 12 and 13. This planar surface has has unit normal $\mathbf{n_r}=(0,0,1,0)$ with respect to the robot coordinate frame, however we require the expression with respect to the camera, therefore we require $\mathbf{n_c}= g_{cr}\mathbf{n_r}$, the plane normal with respect to the camera coordinate frame. The inner product of any point on the plane with the normal is the distance from the plane to the origin (Equation 12). Substituting the perspective equation (Eq.11) into Equation 12 and solving for Z we obtain the expression which relates image coordinates to scene depth for a pixel in terms of its image coordinates and the camera orientation (Equation 13).

$$\boldsymbol{n_c P} = g_{cr}\boldsymbol{n_r P} = g_{cr}\begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = d \quad (12)$$

$$Z = \frac{df}{(n_{ry}v + n_{rx}u + n_{rz}f)} \quad (13)$$

$$n_{ry} = (\sin\theta\cos\phi)$$
$$n_{rx} = -(\cos\alpha\sin\phi\sin\theta + \sin\alpha\cos\theta)$$
$$n_{rz} = \sin\alpha\sin\theta\sin\phi - \cos\theta\sin\alpha$$

The implicit assumption about world planar geometry is totally contained within Equation 13, which allows the assignment of a world coordinate to every image coordinate as described by the inverse perspective transformation (Equation 14). Also note that if the yaw(φ) and roll(α) angles are zero then the above depth (Z) does not depend on *u* -the horizontal image coordinate because the image plane becomes parallel to the ground plane in the X (or equivalently *u*)-direction. In this case, all points in the image plane with common *u*-coordinate will correspond to the same depth. This camera orientation is considered to be nominal.

$$\begin{bmatrix} Z \\ X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{df}{(f\sin\theta - v\cos\theta)} \\ \frac{uZ}{f} \\ \frac{vZ}{f} \\ 1 \end{bmatrix} \quad (14)$$

The above is a nonlinear transformation from the image coordinates *(u,v)* to the world coordinates (X,Y,Z) under the planar assumption. The Z-coordinate must be calculated first (using Eq.13), this value is then used to determine X and Y.

2.3 The Complete Transformation

The sequence of projections and rigid body motions required to generate the predicted motion field are as follows: (1) initial image coordinates (u_1,v_1) are projected onto the ground plane to obtain the corresponding initial world coordinates (X_1,Y_1,Z_1) using Eq. 14, (2) for a given camera displacement (g_{c12}), the coordinates of these world points are recalculated with respect to the displaced camera coordinate frame using Eq.9 , (3) finally these world coordinates are projected back onto the image plane to obtain the displaced

image coordinates (u_1,v_1). These steps are depicted in Fig.4. It is assumed that the pitch angle of the camera is large enough such that the horizon is not visible.

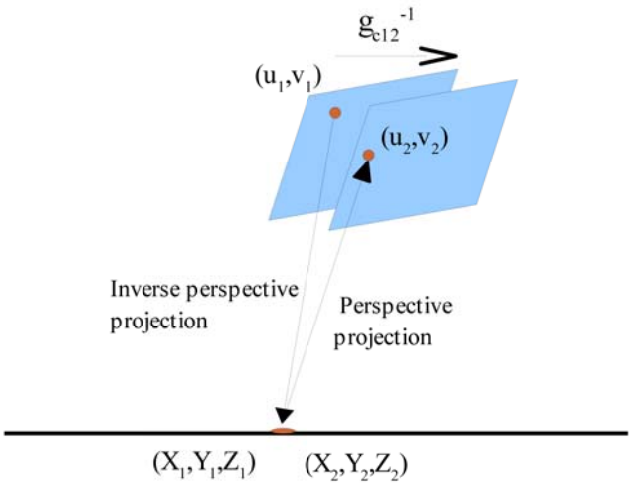


Fig.2.4 Illustration of the steps in computing the predicted motion field.

Applying the aforementioned sequence of operations yields the predicted position of a pixel on the image plane after a given camera displacement. The vector formed by difference between final and initial pixel locations is the predicted displacement vector. As mentioned, the collection of all displacement vectors associated with each pixel form the predicted motion field. Because all other variables are presumed to be static (i.e. camera configuration, and scene geometry) this motion field depends only on the displacement of the camera. Sample motion fields, for various camera motions, are shown in Figure 4. Expected motions for a ground based differentially-steered robot are restricted to three degrees of freedom, they are: translations in the Y_R-X_R plane and rotations as defined by the Z_R-axis in Fig. 3.

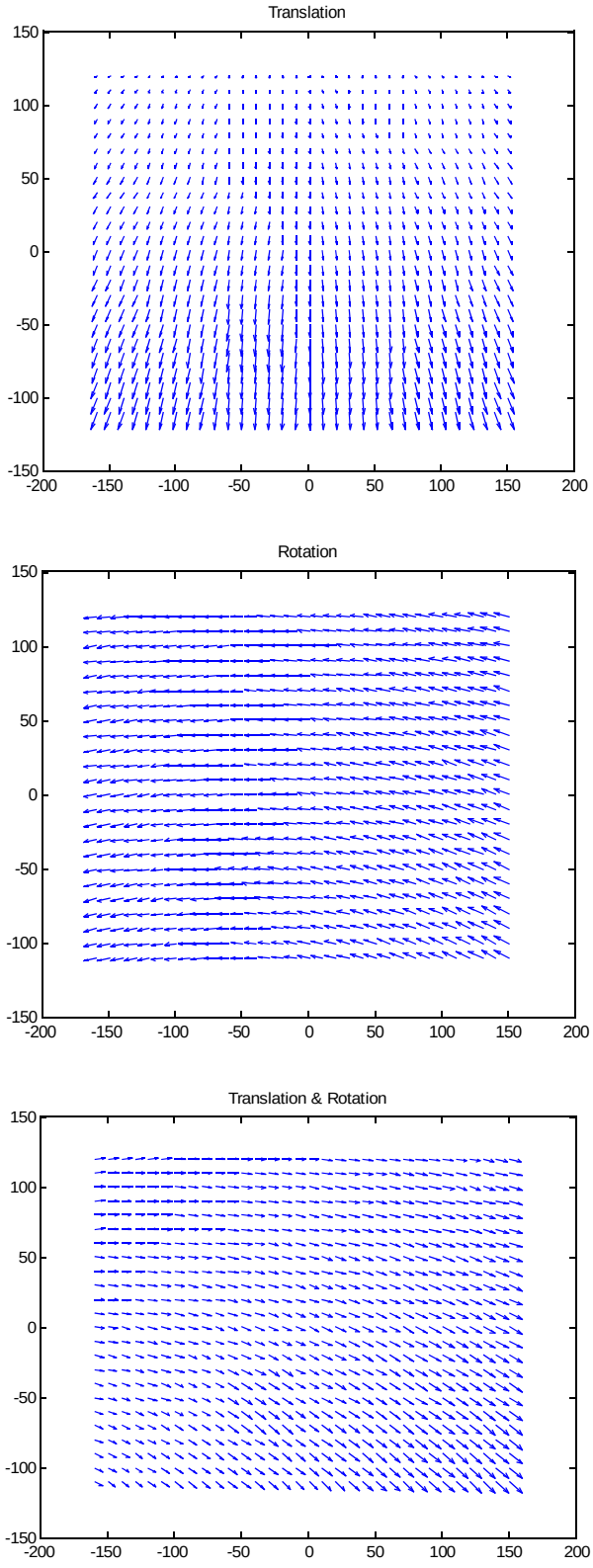


Fig. 2.5 Predicted motion fields for various camera motions.

The predicted pixel coordinates are used to generate predicted images of the scene after a known camera displacement. Because the predicted positions of the image pixels are in general non-integer values, and therefore fall outside the image sampling grid, bilinear interpolation is used to generate the predicted image[3]. That is, the intensity value of a pixel in the predicted image is the weighted combination of the intensity values of its neighboring pixels in the initial image. Bilinear interpolation is described by Equation 15 and depicted in Figure 2.6.

$$I(x,y) \approx \begin{bmatrix} 1-x & x \end{bmatrix} \begin{bmatrix} I(0,0) & I(0,1) \\ I(1,0) & I(1,1) \end{bmatrix} \begin{bmatrix} 1-y \\ y \end{bmatrix} \quad (15)$$

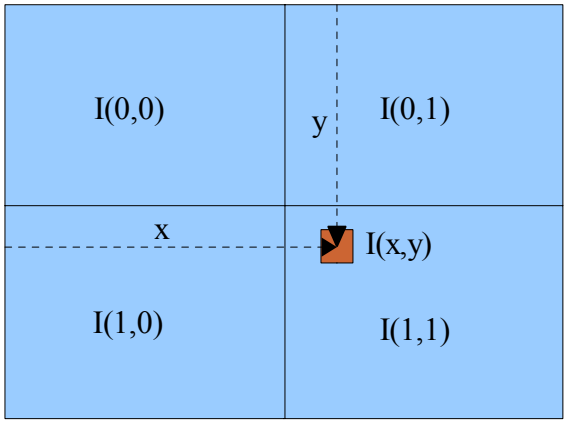
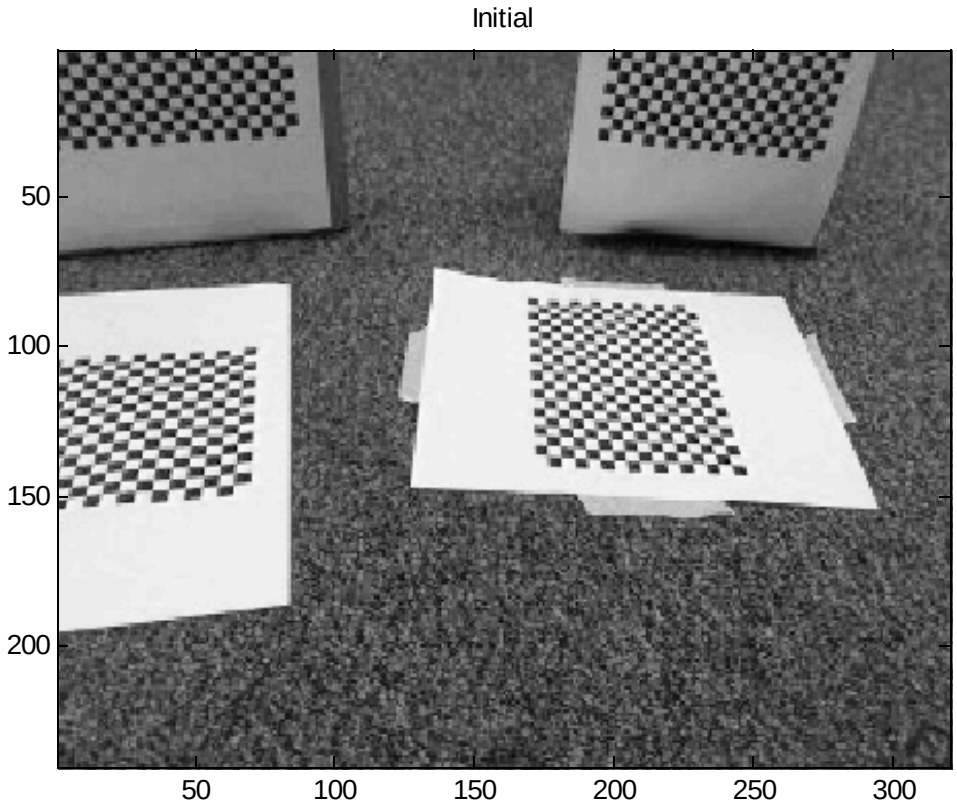
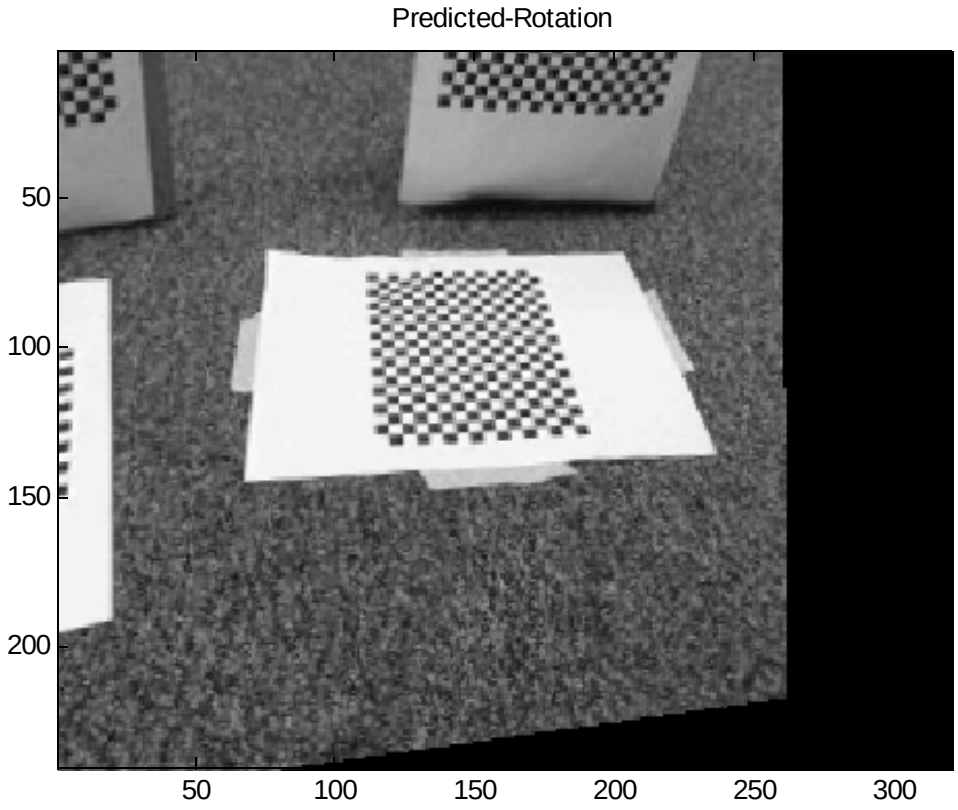


Fig. 2.6 Bilinear Interpolation. I(x,y) is an unknown intensity value between the sampling grid.

Sample predicted scenes are shown in Figure 2.7. These real-world images were captured by a camera on the moving robot. The scene contains two vertical boxes (vertical, with respect to the ground) with checkerboard patterns and two checkerboard patterns on the ground. The predictions were generated using the motion fields in in Figure 2.5 and are based on the initial view of the scene in image (a). Note that the origin in Figures 2.5 and 2.6 is at the center of the image, however in Figure 6 the origin is at the top left corner in accordance with image processing convention. The predicted images above correspond to

typical expected motions of the robot. Image (b) corresponds to a translation of 50mm in the $+Y_R$ direction, image (c) corresponds to a 0.15 radian rotation in the Z_R direction, and image (d) corresponds to a 50mm translation in the $+Y_R$ direction and a rotation of -0.15 radians as defined by the Z_R -axis. Also note that the black regions of the predicted images represent areas where new scene information is entering into view and thus cannot be predicted. The following chapter will describe how these predicted images are used for obstacle detection.





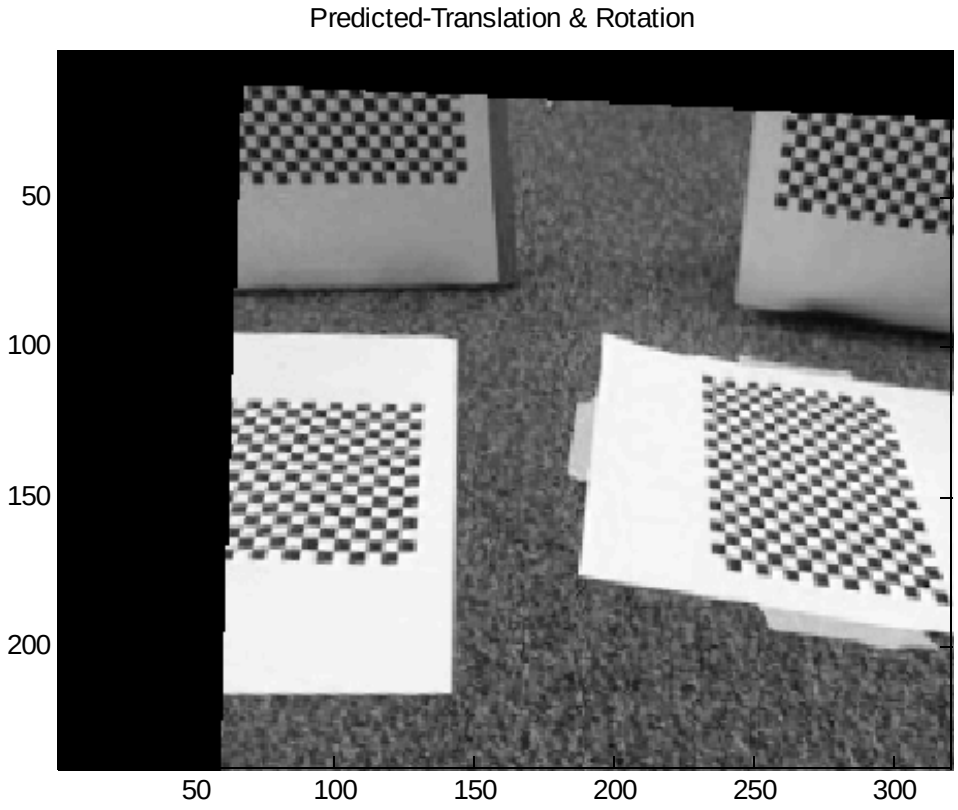


Fig.2.7 Predicted images corresponding to the motion fields of Fig. 2.5

CHAPTER 3

OBSTACLE DETECTION USING APPARANT MOTION

The previous chapter described how to generate predicted images of a static scene observed by a moving camera, under the assumption that the observed scene was a planar surface. This chapter will describe how to use these predicted images for obstacle detection. Recall that an obstacle for a ground based vehicle is defined as an object whose Z_R- coordinate is nonzero, that is, it lies above or below the ground plane.

Section 3.1 will illustrate by means of simulation the underlying theory of this chapter by referring to the predicted motion fields derived in Chapter 2. Section 3.2 will discuss optical flow: a computation which attempts to *measure* (as oppose to predict) relative motion between images. Several practical problems of optical flow will be illustrated. In light of these problems a refinement of the basic approach will be made in Section 3.3 which will discuss the purpose of generating predicted images.

3.1 Determining Sensor Sensitivity through Simulation

The predicted motion fields illustrated in Section 2.3 predict the apparent motion of a planar surface as viewed by a camera. If a *measurement* of the actual motion field observed by the camera is available it is possible to compare the observed motion vectors with their corresponding predictions. For a calibrated camera, those motion vectors which disagree with their predictions must not satisfy the planar assumption and should therefore be considered as obstacles.

To illustrate this idea, consider the Euclidean norm of the predicted motion field corresponding to a pure translation (Fig. 2.5). This results in a smooth surface shown in Fig. 3.1. The vectors with the greatest magnitude occur at the bottom of the image plane. Therefore the top of the surface (red) corresponds to the bottom of the image plane and the bottom (blue) corresponds to the top of the image plane.

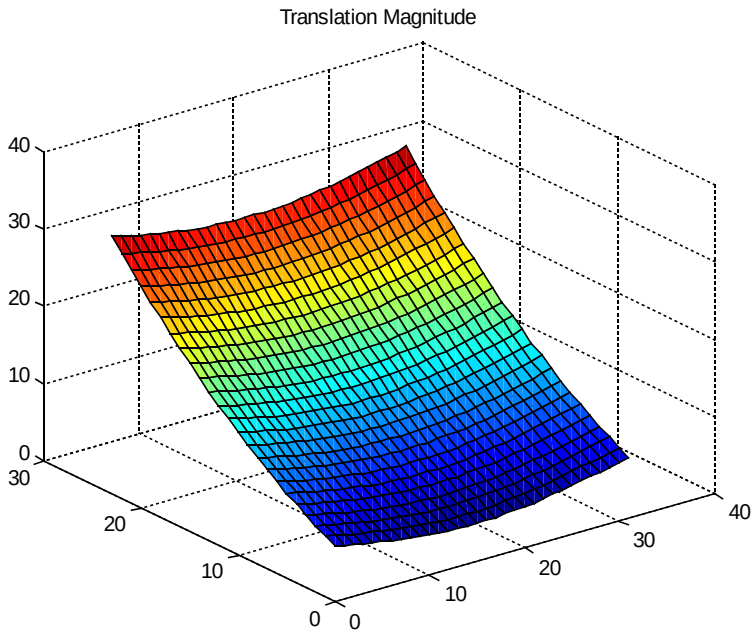


Fig. 3.1 Magnitude of the motion field produced by a forward translation.

Now consider an environment comprised of the ground plane and vertical plane representing an obstacle (Fig. 3.2). Assume that as the camera is translating towards the obstacle that at some instant in time the ground plane occupies the lower half of the image plane, and the obstacle occupies the upper half of the image plane. This is simulated by projecting the lower half of the pixels in the image plane onto the ground plane with unit normal **n** and the upper half onto a plane with unit normal $\mathbf{n}_o$.

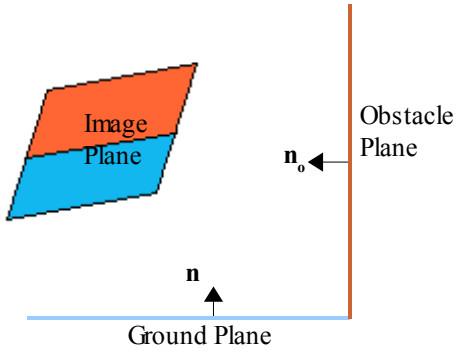


Fig. 3.2 An environment with an obstacle. The top of the image plane is occupied by the obstacle and the bottom is occupied the the ground plane.

The motion field produced by the same forward translation (used to generate Fig. 3.1) in this new environment is shown in Figure 3.3, and its magnitude in Figure 3.4.

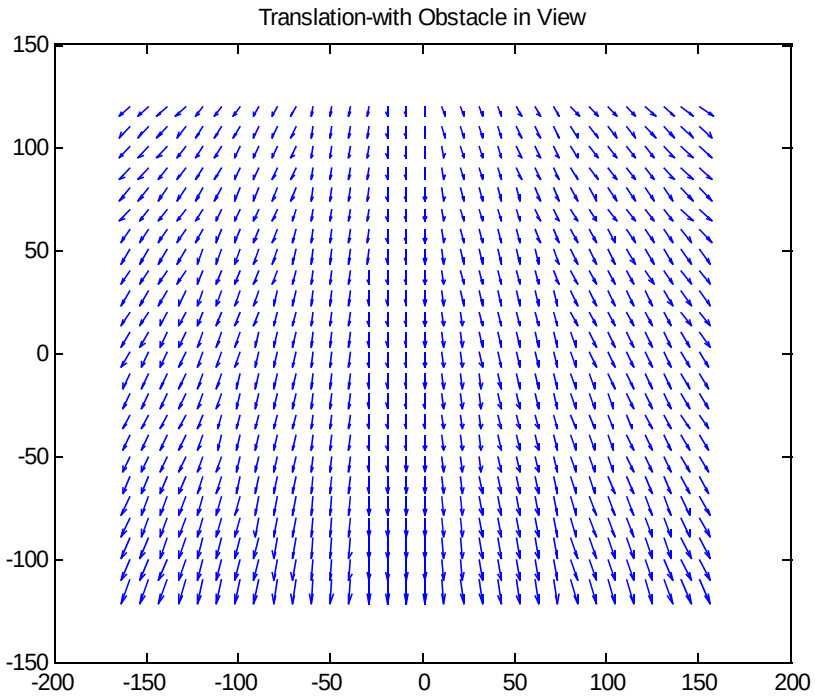


Fig. 3.3 Predicted motion field of the scene in Fig. 3.2 due to a forward translation.

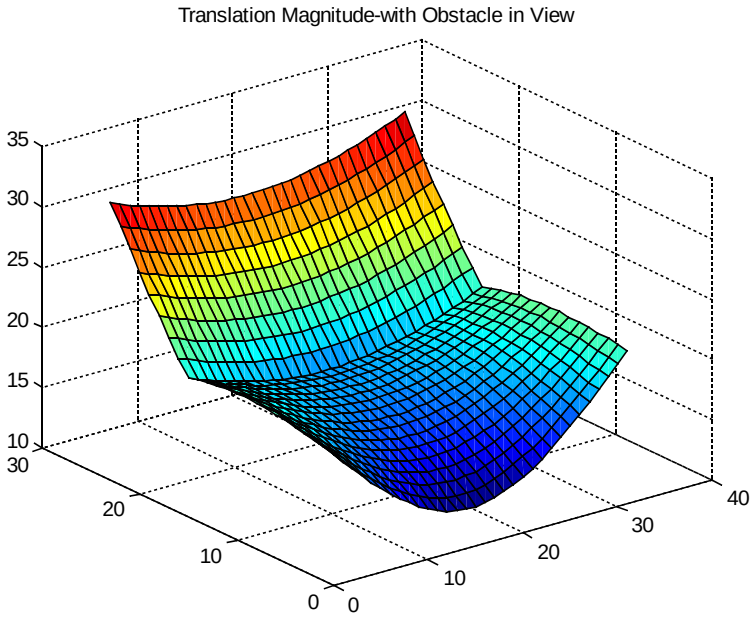


Fig. 3.4 Magnitude of the motion field shown in Fig. 3.3.

The comparison of Figure 3.4 with Figure 3.3 suggests that objects whose geometries do not satisfy the planar assumption will induce deviations from the expected planar motion field. The absolute difference between the two surfaces represents the deviation from the planar prediction and results in the error surface shown in Fig. 3.5. Clearly the error is zero in the lower portion of the image plane.

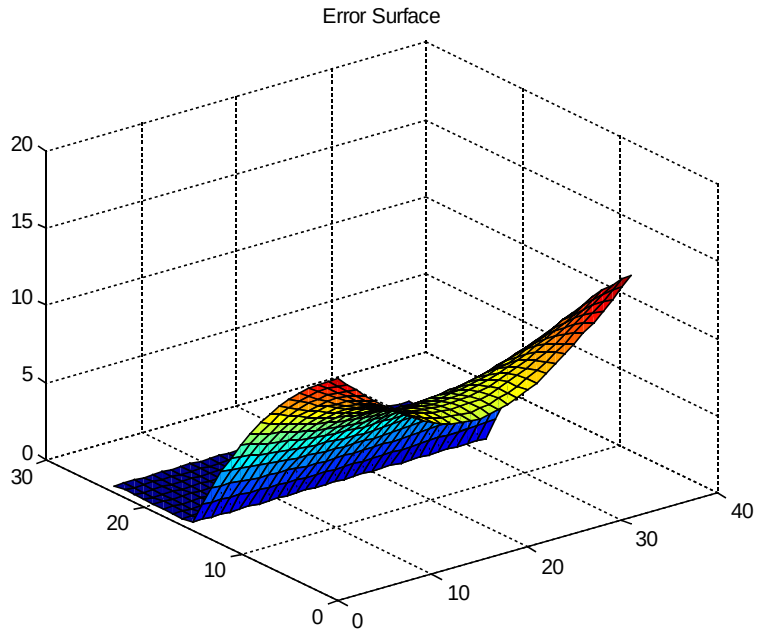


Fig. 3.5 The error surface represents the sensitivity to obstacles of the type depicted in Fig. 3.2.

The error surface displayed in Figure 3.5 reveals the theoretical sensitivity of the proposed approach to the detection of obstacles on the plane with normals perpendicular to the ground plane (walls for example). The greatest deviation from the planar prediction occurs at the top corners of the image plane, therefore the sensitivity to obstacles will be greatest in those regions of the image plane. Of course, different obstacle geometries and camera motions will produce different error surfaces.

3.2 Optical Flow

The approach outlined in the previous section hinges on the assumption that the actual motion field observed by a moving camera is available as a *measurement.* Optical flow is the name given to the approximation of the motion field obtained by measurement of image quantities. This section will outline a popular method, named after Horn and

Schunck, for estimating motion between adjacent frames in a video sequence[4]. The Horn & Schunck optical flow computation will be briefly outlined in the first part of this section, the second part of this section will discuss the shortcomings and limitations of this computation.

The goal of any optical flow computation is to estimate the apparent motion of objects in a video sequence. That is, if two images (I_1 and I_2) are taken at separate instances in time we want to estimate, for each pixel, a two dimensional motion vector which represents the displacement of that pixel in image I_2 relative to its initial location in I_1. The field formed by the ensemble of these vectors is exactly analogous to the motion fields derived in the previous chapter. However, in the previous chapter the motion fields were predicted, now the goal is to estimate the motion field given the two images I_1 and I_2. Assume that an image sequence is a mapping of the form shown in the set relation:

$$\begin{array}{c} I : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R} \\ (u, v) \times t \to intensity \end{array} \quad (16)$$

This relation states that a sequence of images (video) is the mapping from the image coordinates and the instant in time, at which the observation is made, to the pixel intensity. Of course, this set relation assumes continuity in time and space which we will also assume in this section.

It is presumed that the intensity of the objects in the dynamic scene are constant. That is, changes in image intensity are solely caused by the motion of objects and not by any changes in illumination. This assumption implies that if we are able to track a point ($\bar{u}$,v) associated with an object from frame to frame, its intensity would remain constant. This

is expressed by Equation 17, differentiating both sides with respect to time yields Equation 18.

$$I(\bar{u}, \bar{v}, t) = c \quad (17)$$

$$dI(\bar{u}, \bar{v}, t)/dt = I_u \bar{u}_t + I_v \bar{v}_t + I_t = I_u p + I_v q + I_t = 0 \quad (18)$$

In Equation 18 subscripts indicate partial derivatives. The derivative of the tracked point's coordinates with respect to time form the motion vector (*p,q*) for that point. However these are two unknowns with one equation (Equation 18). Therefore an extra "smoothness" constraint is imposed which roughly states that as the motion field is traversed the vectors must vary in a smooth fashion, both in magnitude and direction. A variational problem is formulated where the energy functional takes the form:

$$E(p, q) = \frac{1}{2} \int \int (I_x p + I_y q + I_t)^2 \, dudv + \frac{\lambda}{2} \int \int \|\nabla p\|^2 + \|\nabla q\|^2 \, dudv \quad (19)$$

That is, we wish to find a vector field described by horizontal component (*p*) and vertical component (*q*) that minimizes the functional in Equation 19, where the integrals are taken over the entire vector field. The smoothness constraint is introduced into the functional in the form of the square of the norm of the gradient of the vector field components *p* and *q*. The smoothness constraint can be relaxed by reducing the value of the parameter λ.

Minimizing the integral on the left will result in the minimization of the optical flow constraint (Eq. 18), and minimizing the integral on the right will ensure a smooth vector field. The application of the Euler-Lagrange equation to the functional above yields the iterative gradient descent solution shown in Equation 20. This equation is discretized and used to iteratively solve for the optical flow components *p* and *q* [5].

$$p^{(k+1)} = -\nabla_u E^k = -(I_u p^k + I_v q^k + I_t) I_u + \lambda (p^k_{uu} + q^k_{vv})$$
$$q^{(k+1)} = -\nabla_v E^k = -(I_u p^k + I_v q^k + I_t) I_v + \lambda (p^k_{uu} + q^k_{vv}) \quad (20)$$

The equation above implies that all motion information is represented by the partial derivatives of the the image. This leads to incorrect motion vectors in two cases:

1. Because derivatives capture local changes in intensity, it should be expected that the optical flow computation will fail for large spatial displacements.

2. Moving areas of the image with constant intensity will not produce any apparent motion. This is a special case of what is called the aperture problem[4].

The motion field generated by the optical flow calculation is said to be correct if the second image (I_2) can be obtained by warping the first image (I_1) according to the motion field obtained from the optical flow computation. To illustrate this, optical flow between synthetic images is computed next.

Figure 3.6 shows two synthetic images. The initial image (a) contains a black square (intensity zero) on a white background (intensity 255). The second image (b) shows the same square displaced by one pixel in both the horizontal and vertical directions, corresponding to a total displacement of about 1.414 pixels. Figure 3.7 shows the motion field (a) obtained from the optical flow equations (Eq.20) with $\lambda=\sqrt{2}$ after 10 iterations. This motion field is used to warp the initial image by means of bilinear interpolation discussed in Section 2.3. Ideally, the warped image(b) should match the displaced image

(Fig.3.6 (b)), however several discrepancies between the two images are present. The black lines on the edges of the image represent portions of the image where new image information is entering and therefore cannot be determined. The nonzero (on the order of 10^{-12} pixel displacement) optical flow in the the portions of the image where intensities have not changed are due to the smoothness constraint imposed on the optical flow field. Other discrepancies are due to motion field and interpolation errors.

Next consider a large displacement of the same object. Figure 3.7 shows the square displaced by 10 pixels in both the horizontal and vertical directions, corresponding to a total displacement of about 14.14 pixels. Warping the initial image (Fig. 3.6 (a)) according to the obtained motion field yields the warped image (c). Clearly, the warped initial image does not match the displaced image for large displacements.

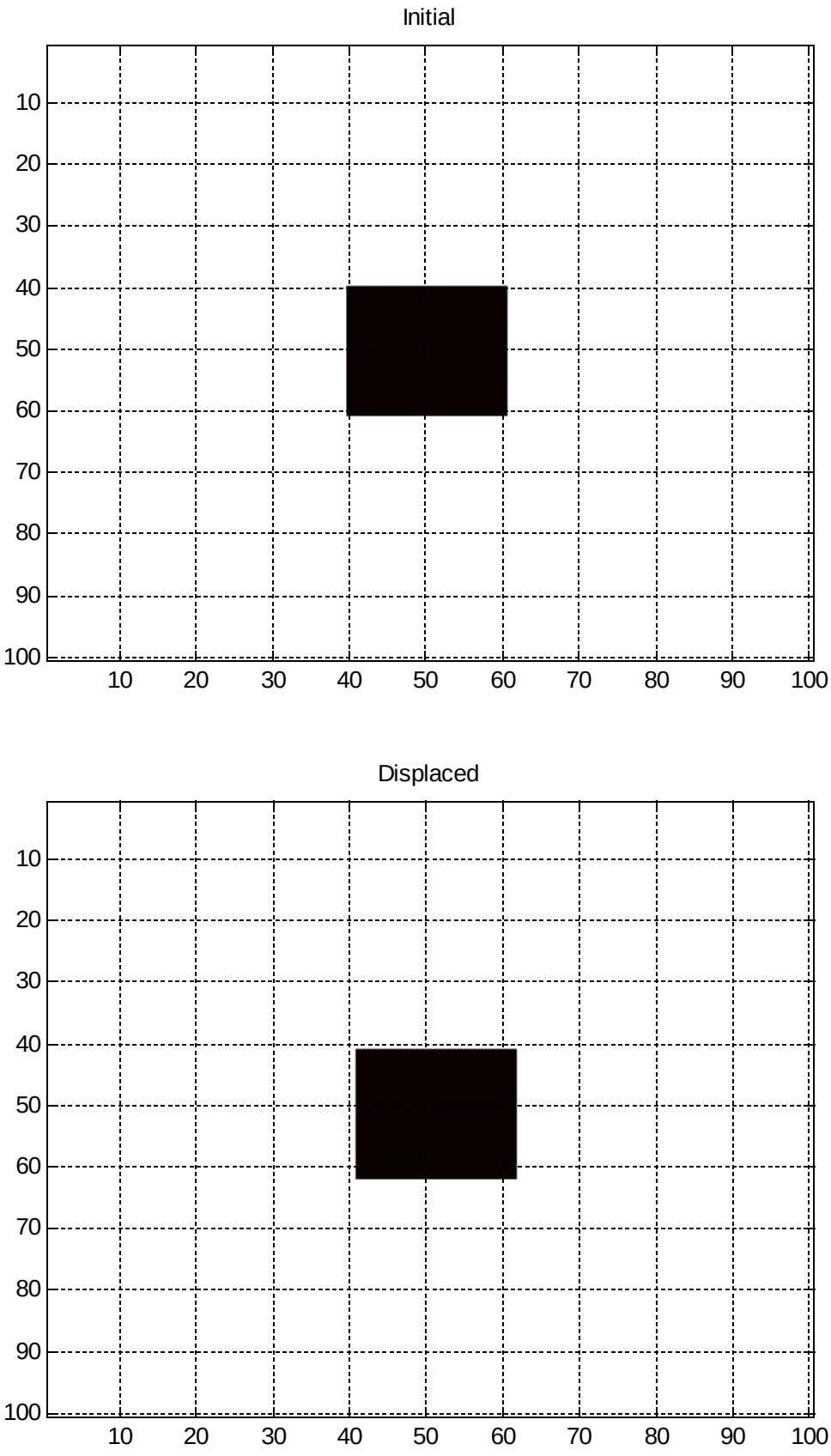
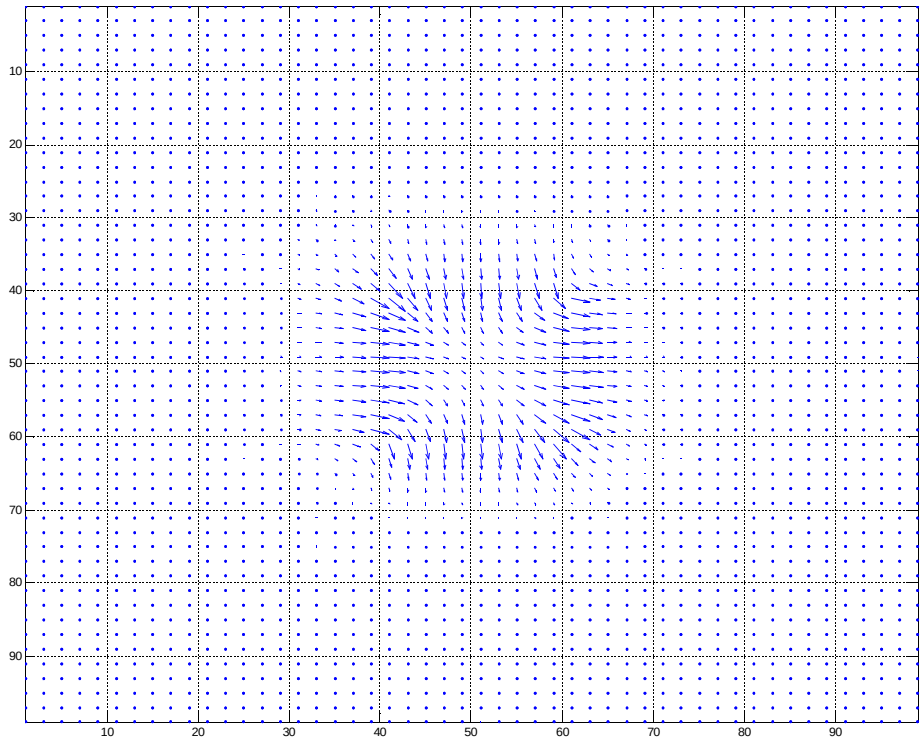


Fig. 3.6 Initial(a) and Displaced(b) Images Corresponding to a Small Displacement



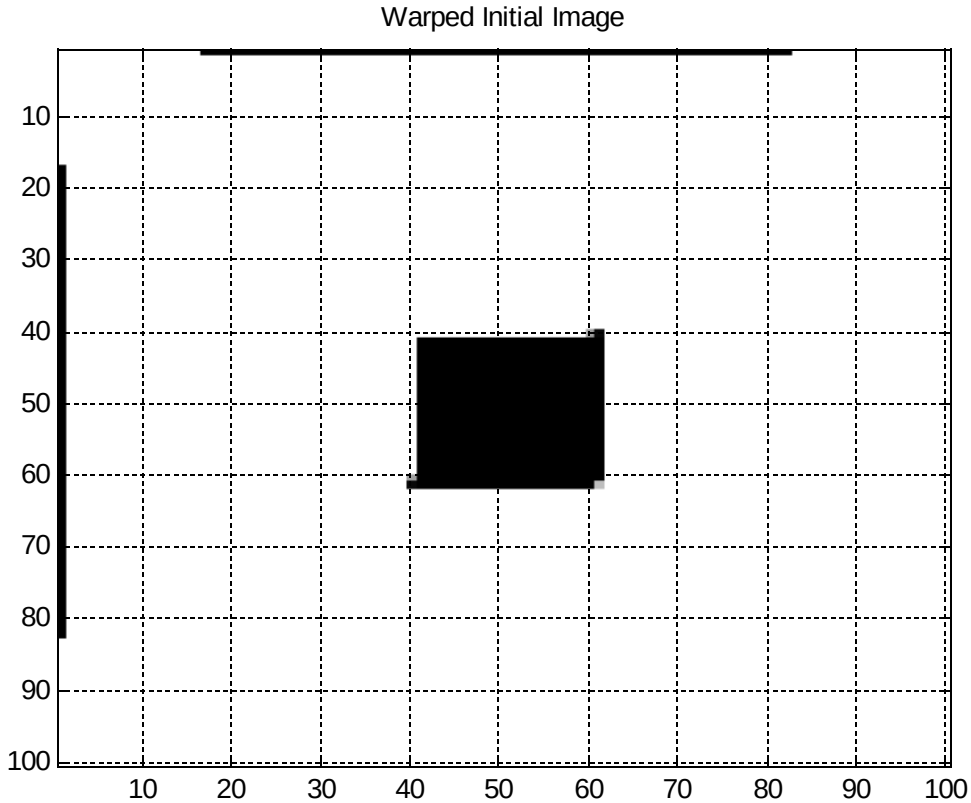


Fig. 3.7 Motion Field(top) for Small Displacement and Corresponding Warped Image(bottom).

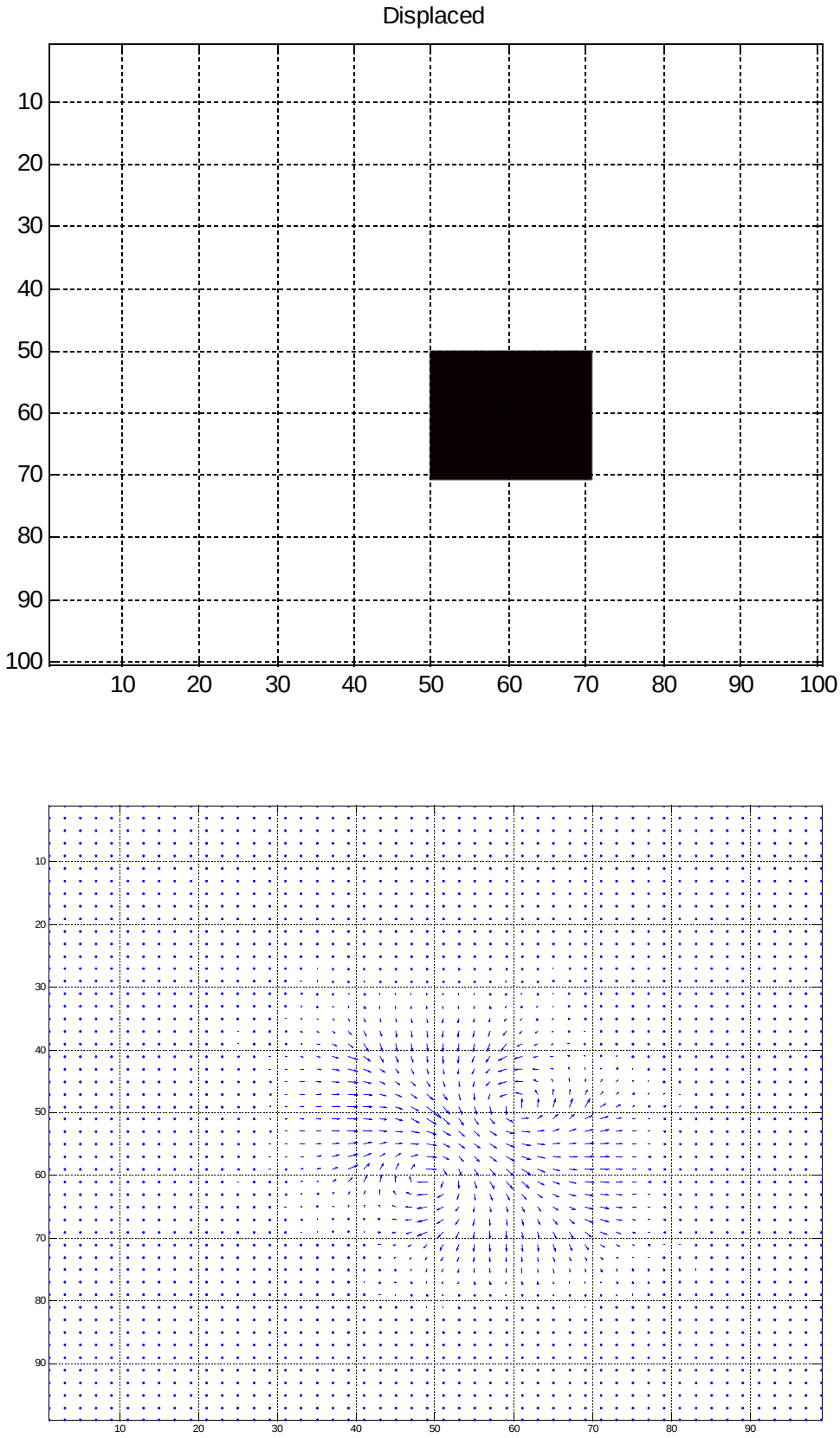


Fig. 3.8 The displacement image and the corresponding motion field for a large displacement.

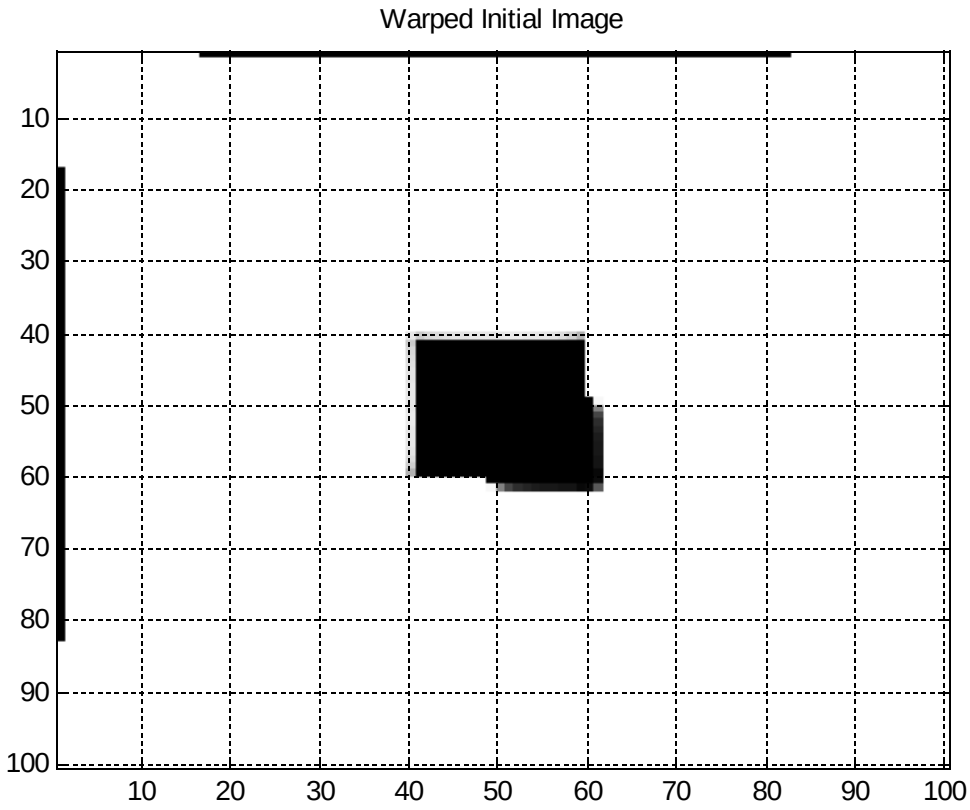


Fig. 3.9 The warped initial image corresponding to a large displacement.

3.3 Reducing Inter frame Motion

In light of the shortcomings of the optical flow computation for large displacements illustrated in the previous section, this section will discuss the reasoning behind generating the predicted images discussed in the previous chapter.

Assume that images I_1 and I_2 are temporally adjacent frames in a video sequence captured by a camera mounted on the moving robot. It should be mentioned that the robot displacement between I_1 and I_2 is generally unrestricted and can be large. Therefore the apparent displacement of objects between I_1 and I_2 may also be large.

The planar motion field prediction may be used to detect non-planar objects in two different approaches:

1. Estimate the motion field between adjacent frames (I_1 and I_2) in a video sequence and compare this measured field with that of the predicted motion field based on the planar assumption derived in Chapter 2. Those measured vectors that do not conform to the planar prediction must correspond to world points that do not lie on the ground plane and should therefore be considered as obstacles.

2. Warp the initial image (I_1) according to the predicted planar motion field (F_p) to generate a "predicted image" (I_p), and compare I_p to I_2 (the actual image taken from the new position). Those portions of I_2 which disagree with the prediction I_p must correspond to world points that do not lie on the ground plane and should therefore be considered as obstacles.

Note that approach (1) relies on the comparison of motion fields (as in Section 3.1) and approach (2) relies on the comparison of images. Motion field comparison is generally straightforward. Similarity between corresponding predicted and measured motion vectors can be quantified by computing their inner product, or by computing the error surface discussed in Section 3.1. Conversely, a good measure of similarity between images cannot be their absolute difference as illustrated next.

Consider an initial image (I_1) shown in Figure 2.7 (a). Suppose it is known that the camera underwent a forward displacement of 50mm between I_1 and I_2. Given this displacement, it is possible to generate the predicted view of the scene (I_p) at the new position under the planar assumption, as described in Chapter 2. Following approach (2)

outlined above, it is necessary to compare I_p (predicted image) to I_2(actual image) to determine which portions of the image violate the planar assumption which was used to generate I_p. The predicted image is shown in Figure 3.10 (a) and the actual image taken at the location is shown in Figure 3.10 (b).

To have a valid comparison between the two images portions of the world scene which have entered or left the camera view must be ignored, this is the reason for the black boundary introduced around both images. Computing the absolute difference between the predicted and actual images yields the error image in Figure 3.11 (a). Computing the optical flow between the same images and plotting the magnitude of the resulting motion field(called the "error motion field") leads to the image shown in Figure 3.11 (b).The purpose of comparing the predicted and actual images is to identify the non-planar objects. With this criteria in mind, the optical flow computation is a better measure of "distance" between the predicted and actual images. The pixel-wise image difference will have large values wherever the predicted and actual images differ in intensity. Therefore even a small misalignment (such as the one that occurs on the lower right checkerboard pattern), leading to large intensity differences will lead to large values on the error image. In other words, the optical flow computation measures the severity of misalignment of the predicted and actual images, irrespective of the actual intensity values at particular locations.

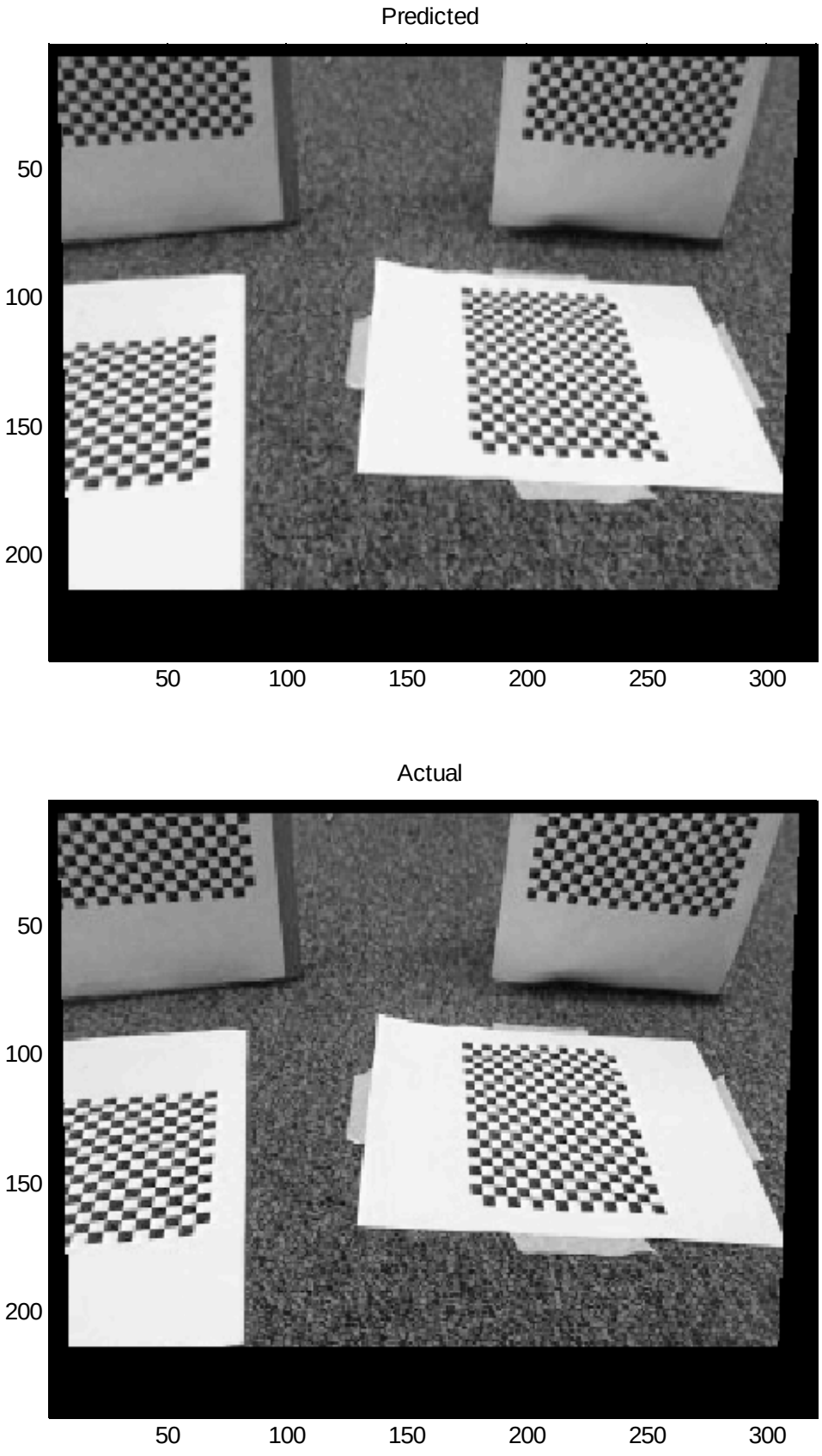


Fig. 3.10 Top (a): Predicted Image. Bottom (b): Actual Image

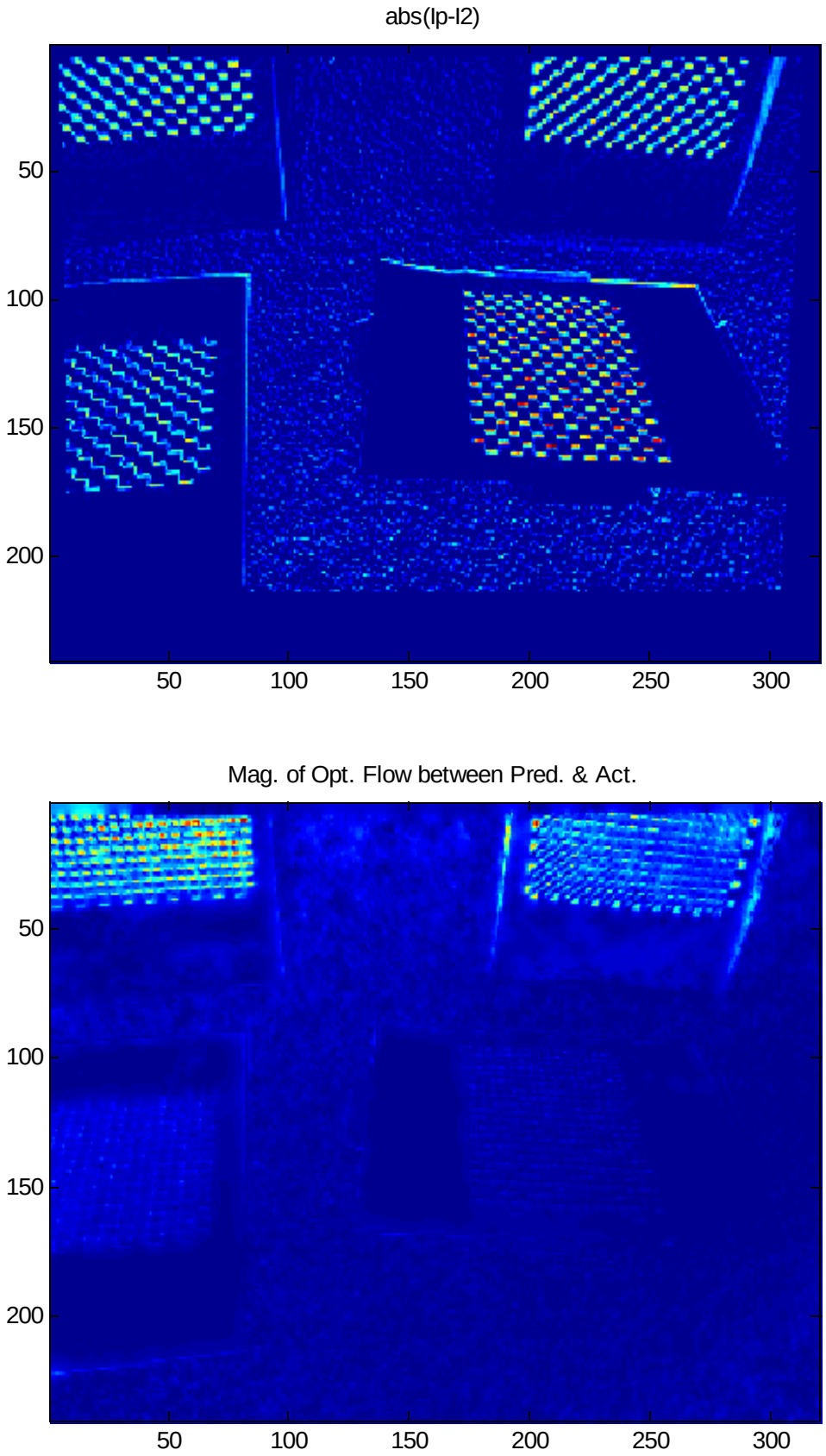


Fig. 3.11 Top(a) Absolute value of the difference between Ip and Ia. Bottom(b) Magnitude of the optical flow between Ip and Ia.

To illustrate the fundamental difference between approach 1 and 2, consider an image space where similar images lie close to each other, call it spatial-intensity space. That is if the images have similar intensity values at corresponding locations then they are metrically close in this space. The metric can be defined as the magnitude of the motion field obtained by an optical flow computation. The dimension of this space is the product of the horizontal and vertical resolutions of the image, however for convenience the images are drawn as points in two-dimensional Euclidean space.

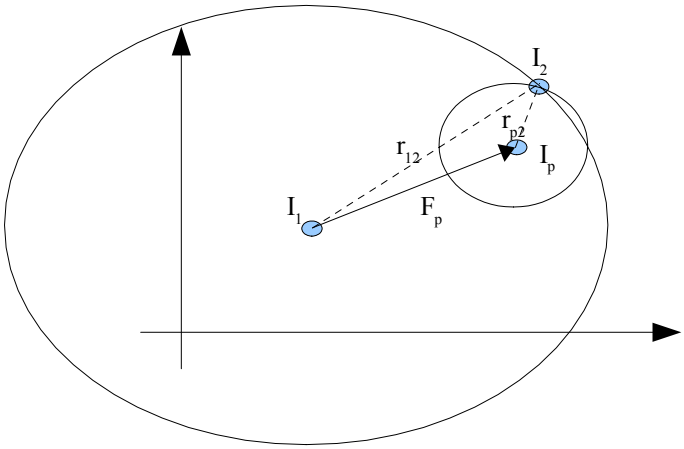


Fig. 3.12 Illustration of the image space and relative distances between images. The predicted motion field (F_p) is used to warp I1 to generate Ip.

Figure 3.12 illustrates the difference between the two approaches. Approach 1 calls for an optical flow computation between the initial (I_1) and final (I_2) images. Because these images are taken at separate instances by a moving camera they are generally "far apart" with respect to the metric described above. Let the distance between I_1 and I_2 be called r_{12}. The predicted motion field (F_p) is used to interpolate a predicted image I_p, let the distance between the predicted image and the final image be r_{p2}. We will not attempt to formulate a general proof that $r_{p2}<r_{12}$, for all scene geometries and camera motions. However,

generally this was observed to be true . This means that the optical flow computation between I_p and I_1 will generally be more reliable compared to the one between I_1 and I_2. Essentially, generating I_p amounts to making an initial guess on the flow field between I_1 and I_2, the guess being that the flow corresponds to that of the ground plane. Discrepancies in this prediction are revealed by computing the flow field between I_p and I_2 These discrepancies occur at the image coordinates corresponding to non-planar objects. Approach 2 solves the large displacement problem of the optical flow calculation by effectively reducing the distance (i.e. the amount of motion) between the images.

Figure 3.11 (b) illustrates that certain features (corners and edges) on both vertical boxes have been detected as obstacles. However recall that one of the shortcomings of optical flow computation is that featureless areas (untextured areas without edges or corners) will produce no apparent motion. Indeed this is confirmed in the figure; lower portions of the vertical boxes (devoid of the checkerboard pattern) produce very small motion vectors even though they do not lie on the plane. Also note that the magnitude of the error motion field decreases as we observe world points close to the ground plane (such as lower portions of the box), this is simply because objects close to the ground plane have similar apparent motion as the ground plane. The aforementioned issues will play a major role in obstacle segmentation, to be discussed in the next chapter.

CHAPTER 4
OBSTACLE SEGMENTATION

So far we have developed a framework for the detection of points that lie above or below the ground plane. We have defined these points to be obstacles for a unmanned ground vehicle. These points may be part of some larger three-dimensional object. In this case, we wish to segment the entire object. For example, consider Figure 4.11 (b). This figure it would be desirable to generate a segmentation for both vertical boxes.

In the introductory chapter we have discussed the difference between what we call detection and segmentation. In this chapter we will develop a segmentation procedure which attempts to generate a segmentation for the entire obstacle object. Section 4.1 will describe the underlying assumptions used to develop the procedure. Section 4.2 will describe the segmentation algorithm, namely Bayesian region growing. Section 4.3 will describe how detection information is used to initialize the segmentation procedure. In Section 4.4 we will discuss how temporal integration can be used to generate more accurate and reliable segmentations. Finally, in Section 4.5 we will present an approach for calculating the distances to detected obstacles using assumption 4 of Section 1.2.

4.1 Overview of Segmentation Approach

The procedure described in the previous chapter detects salient non-planar feature points on obstacle objects. In this section we will see how these points (called “seed pixels”; the reason for this name will become clear later in this chapter) can be used to segment the

entire obstacle object. The criteria for selecting seed pixels will be described in detail in Section 4.2, for now we simply assume that seed pixels correspond to the most salient feature points of non-planar objects. We further assume that the seed pixels form a representative sample of the color statistics of the obstacle object. The latter assumption validates the use of the region growing segmentation algorithm initialized with the seed pixels. The region growing segmentation algorithm essentially detects connected image regions with similar color distribution, starting in some initial region with initial distribution.

An example of obstacle segmentation using region growing is shown in Figure 4.1. Image (a) shows the actual observed scene which contains a vertical obstacle (wireless router with antenna) atop of a flat sheet of paper, the background is gray carpet. Seed pixels used to initialize region growing are shown in red on the predicted image. The segmentation was done on the predicted image, the reason for this will be discussed in the next section. Note that the seed pixels are clustered around the upper edges of the obstacle. The upper edges are the most salient features which produce the greatest violation of the planar assumption used to generate the predicted image.

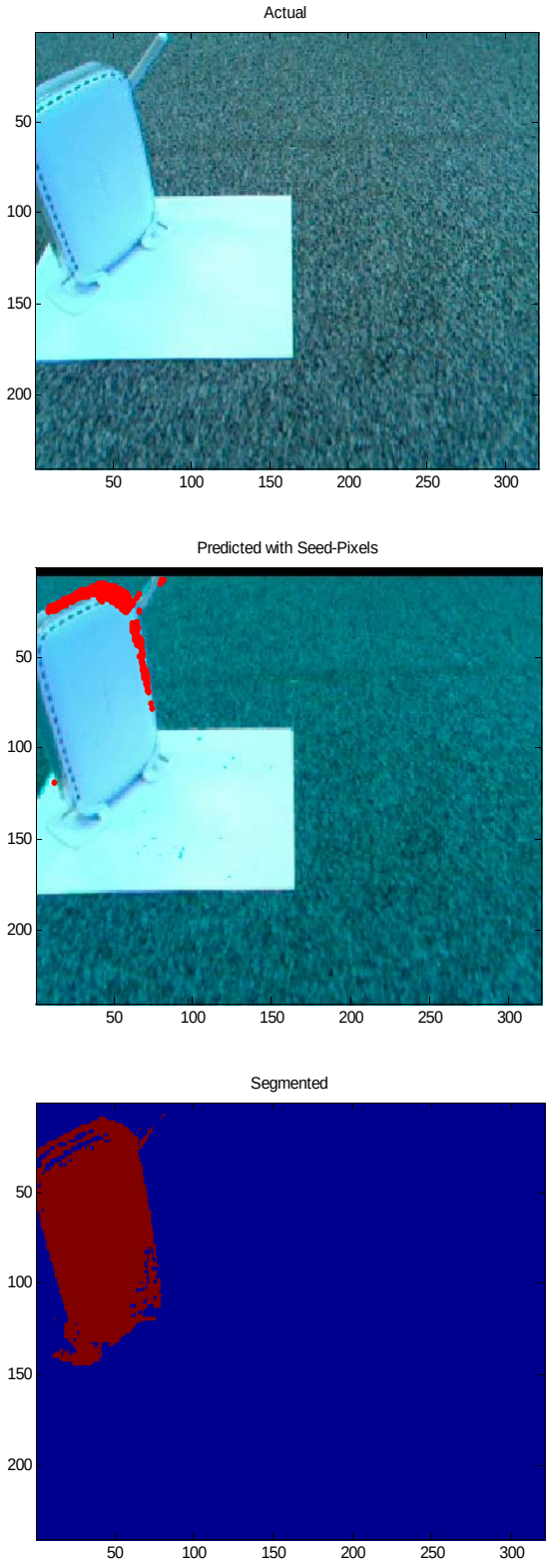


Fig. 4.1 Top (a): actual scene image. Center (b): predicted scene, with seed pixels in red which were used to initialize the region growing algorithm. Bottom (c): Segmentation result of region growing.

4.2 Bayesian Region Growing

In this section we will describe a segmentation algorithm, namely Bayesian region growing[6][3]. This segmentation algorithm returns a connected region in the image with a common color distribution. The algorithm is initialized with a set of pixels believed to be inside the object of interest, we called these "seed pixels". Seed pixel selection will be discussed in greater detail in the next section.

Suppose the color distribution of some object is assumed to be Gaussian, with mean vector $\boldsymbol{\mu}$ and covariance matrix Σ. We assume that the image has three color channels, therefore the mean vector is three-dimensional and the covariance matrix is a three-dimensional, square, positive symmetric matrix. Furthermore, assume that the image can be divided into a discrete number of classes (objects) each describes by its unique mean vector and covariance matrix. Then the probability density function for this multivariate Gaussian distributions is given by Equation 21, which returns the probability of a pixel having intensity value $\mathbf{v}=(v_1, v_2, v_3)$ given that it belongs to class c described by mean vector $\boldsymbol{\mu}_c$ and covariance matrix Σ_c. This distribution is simply the normalized color histogram of the object (class), however we model the histogram as having a Gaussian distribution.

$$Pr(V_i=v|C_i=c)=\frac{1}{(2\pi)^{(\frac{3}{2})}|\Sigma_c|^{(\frac{1}{2})}}\exp(\frac{-1}{2}(v-\boldsymbol{\mu}_c)^T\Sigma^{-1}(v-\boldsymbol{\mu}_c)) \quad (21)$$

By applying Bayes rule with homogeneous prior an expression for the probability that a pixel with intensity value $\mathbf{v}$ belongs to class c is obtained in Equation 22. Where the summation is taken over all the classes (number of objects in the image).

$$Pr(C_i = c | V_i = v) = \frac{Pr(V_i = v | C_i = c)}{\sum_{\gamma} (Pr(V_i = v | C_i = \gamma))} \quad (22)$$

Equation 22 allows us to make a decision whether to include a pixel in a certain class or not. The mean and covariance of classes need not be static and can be updated as more pixels are included in the class.

Starting from some initial seed pixel the region growing algorithm checks the 8-point neighborhood of the pixel and determines if any of it's neighbors should be included in the class. The decision to include a pixel is made if the probability of that pixel belonging to the class exceeds some threshold probability. If a pixel is included in the class its neighbors are checked and the procedure repeats in a recursive manner until all the neighbors of the included pixels have been checked or are included in the class. In this manner, a group of seed pixels belonging to some object can be used extract a connected region over the entire object as depicted in Figure 4.1.

4.3 Initialization

The region growing algorithm described in the previous section requires that the seed pixels used to initialize the algorithm satisfy two criteria, namely: (1) seed pixels must be located inside the boundaries of the object of interest, and (2) seed pixels are a representative sample of the color statistics of the object. Seed pixels are selected as those locations in the image which result in the largest disagreement with the planar prediction, as measured by optical flow between the predicted and actual images. In this section problems regarding seed pixel selection will be addressed which occasionally violate the

aforementioned assumptions.

Ultimately seed pixels are selected by thresholding the magnitude of the optical flow between the predicted and actual images. However before thresholding spatial dependency is introduced by computing the local mean (blurring)[3] and adding to it the original. Introducing spatial dependency reflects the assumption that violations are caused by relatively large objects and therefore must occur in spatially coherent groups. This procedure is described by Equation 23, the result of which is shown Figure 4.2. In the equation below * denotes two-dimensional convolution, and O.F. denotes the optical flow computation.

$$\left\|O.F.(I_p, I_a)\right\|_B = \left\|O.F.(I_p, I_a)\right\| + \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} * \left\|O.F.(I_p, I_a)\right\| \quad (23)$$

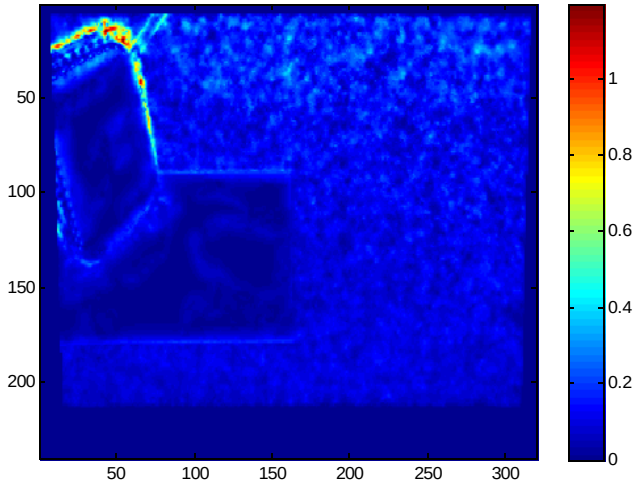


Fig. 4.2 Smoothed version of the magnitude of the optical flow between the predicted and actual images.

Seed pixels are selected by applying a threshold to the image in Figure 4.2. Applying a threshold of 0.5 followed by the morphological close operation [3] results in the binary

seed pixel locations depicted in Figure 4.1 (b). The corresponding image locations of these seed pixels are used to extract the statistics used for region growing of the seed pixel cluster.

It is assumed that seed pixels fall are located inside the boundaries of the non-planar objects. Generally this assumption is invalid. This particular problem which we call “seed pixel alignment” is a special case of the aperture problem faced in motion segmentation [7][8]. The problem is illustrated in Figure 4.3. The predicted and actual images of a ring-shaped obstacle are shown, with the seed pixels overlaid in red. The magnitude of the optical flow between the predicted and actual images is shown in the same figure. Two distinct clusters of seed pixels are associated with each of the obstacle's non-planar most edges. However, the top cluster aligns with the obstacle's edge in the predicted image and the bottom cluster aligns in the actual image. For instance, if seed pixels are assumed to align with the obstacle in the actual image when in fact they align in the predicted, then the region growing algorithm will be performed on the wrong object, mostly likely the background. Because optical flow measures motion simply based on changes in image intensity there is no general solution for determining whether a given cluster of seed pixels aligns with the obstacle in the predicted or actual image based on optical flow alone.

Additional information is required to determine whether a given cluster of seed pixels corresponds to an obstacle on the actual or on the predicted image. One approach is to make further use of an earlier assumption, namely that obstacles differ in color from the background. If the histogram of the ground plane is initially known then it is possible to

perform Bayesian background segmentation and exclude those seed pixels which fall onto planar textures for region growing. However, new planar textures may be encountered as the camera moves through the environment, therefore the histogram corresponding to known planar objects must be updated during operation. The test for planarity may be used as the rule for updating valid ground textures. That is, if a given image texture has satisfied the planarity constraint it's color statistics are included in the ground histogram. If however, a given texture has failed the planar constraints it's color statistics are removed from the ground histogram. This approach is called background-masking.

The approach is depicted in Figure 4.4 for an image sequence. The initial scene contains two principal ground textures: the gray carpet, and the unrolled white masking tape. Assume that the ground histogram is initialized to include only the gray colored carpet. Background masking applied to the first image (a) yields the corresponding binary image (b). The white areas of the image represent textures which may be used for region growing (if seed pixels are initialized on them). After testing the planarity of the scene it was determined that the unrolled masking tape satisfied the planar constraint, therefore its histogram was added as a ground texture (d). In the last image however, a white colored obstacle with similar color distribution as the unrolled tape has entered into view. Therefore all objects in the scene of similar color (this includes the unrolled tape) are now considered as potential obstacles and are valid textures for region growing. This approach uses a single histogram to describe the ground texture of the entire image, however multiple histograms representing local ground textures are more suitable.

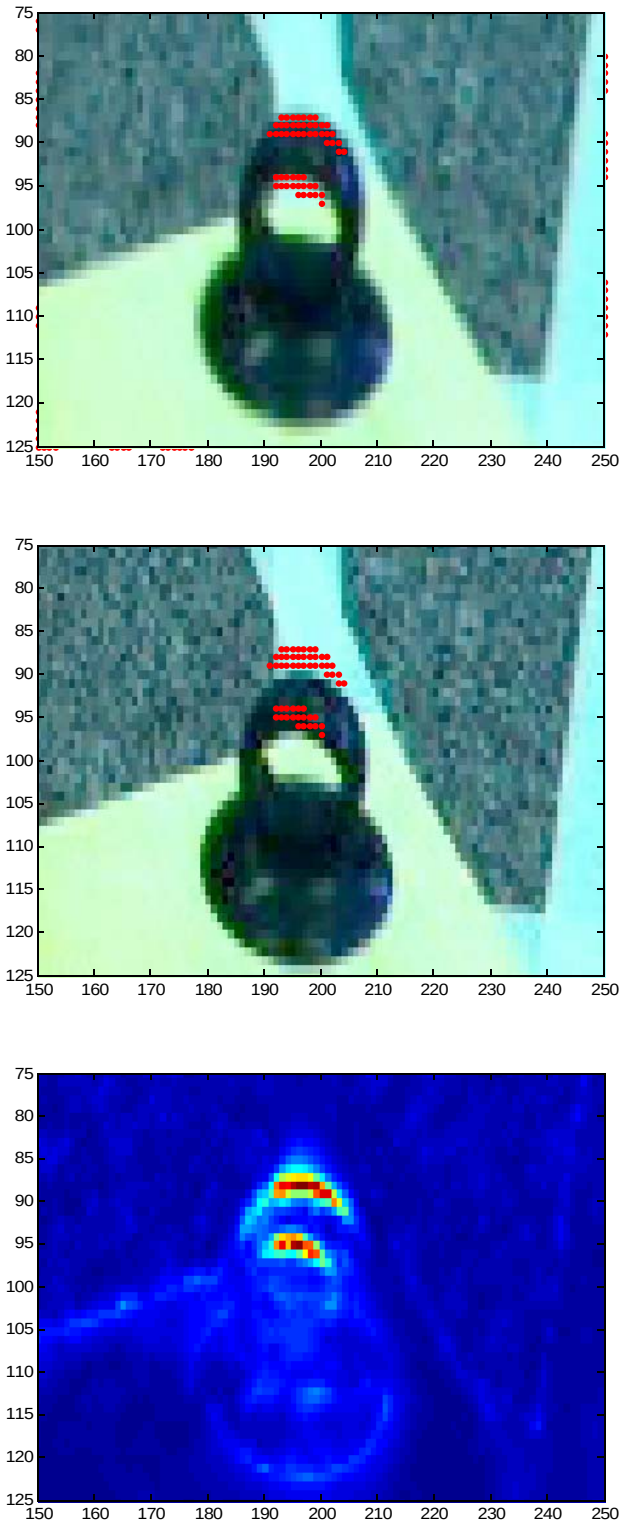


Fig. 4.3 Depiction of the seed pixel alignment problem. An obstacle in the predicted image (a) with seed pixels overlaid in red. The same obstacle in the actual image (b) with seed pixels also overlain in red. The magnitude of the optical flow (c) between images (a) and (b).

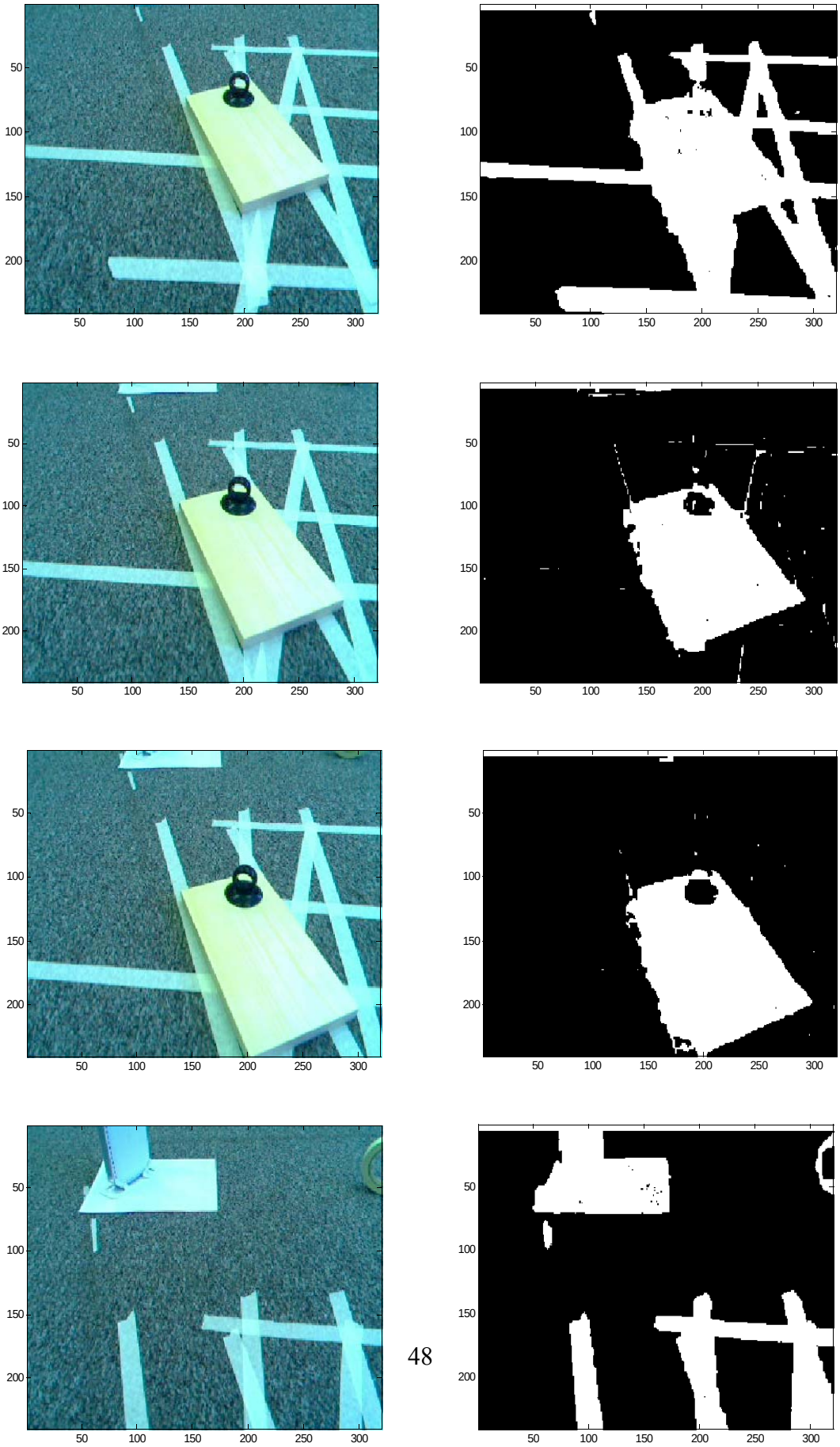


Fig. 4.4 Ground-Masking Illuminates all Seed Pixels which Fall on the Background

4.4 Temporal Integration of Segmentations

Although the final obstacle segmentation is a binary image, it should be reemphasized that it is the result of thresholding the image obtained by executing the Bayesian region growing algorithm on seed pixels which are believed to lie within the boundaries of an obstacle. Bayesian region growing assigns, to each pixel, the probability of being part of an obstacle, which results in a probability field for the entire image. These probabilities are assigned based on color similarity to known nearby obstacle pixels (initially seed pixels).

Obstacle segmentations are not temporally independent. If an obstacle was segmented in the previous frame and is still in the camera's view, it should be expected that it will be segmented again. If the obstacle is not segmented again, it is likely that the previous segmentation was an error or the obstacle is very close to the ground and is only occasionally detected. Subsequent segmentations of the same obstacle will be in apparent motion from frame to frame in the observed image due to the motion of the camera. With reference to the arguments presented in Chapter 2, the apparent motion of the the obstacle in the image plane is a function of the motion of the camera and the three dimensional geometry of the obstacle. However because the geometry of the obstacle is unknown, we approximate its apparent motion with the motion of the ground plane derived in Chapter 2. More elaborate approximations can be used, for example it may be more reasonable to assume that the obstacle has a geometry corresponding to a vertical plane, perpendicular to the ground plane. Using this approximation, the obstacle's position in the current frame may be predicted by warping its previous segmentation forward. Because segmentations

are performed independently of one another, the alignment of the predicted position of the obstacle, based on the previous segmentation, with its current segmentation reinforces the likelihood of a correct segmentation. In this manner the estimate of the location of obstacles can be refined as the obstacle is detected in subsequent frames. Because the Bayesian region growing segmentation procedure returns a probability field, alignment of the predicted and current probabilities amounts to independent observations of the likelihood that a given pixel is an obstacle. These likelihoods are combined using an arithmetic mean for simplicity (geometric mean may be more appropriate). Equation 24 states that the probability of a pixel belonging to an obstacle at the n+1-frame is the weighted average of the previous probability, warped forward according to the planar prediction, and the current segmentation, as given by Bayesian region growing.

$$I_p(n+1)=\frac{(g_{plane}I_p(n)n+I_{seg})}{(n+1)} \quad (24)$$

Segmentation results using this temporal integration scheme are shown in the image sequence in Figure 4.5. This image sequence contains two obstacles, a ring-shaped roll of tape on right and a box-shaped wireless router on the left. Blue represents low confidence (probability) that the pixel is part of an obstacle, red represents high probability that a pixel is an obstacle. As the obstacles are segmented in more and more frames, the confidence in these detections is increased.

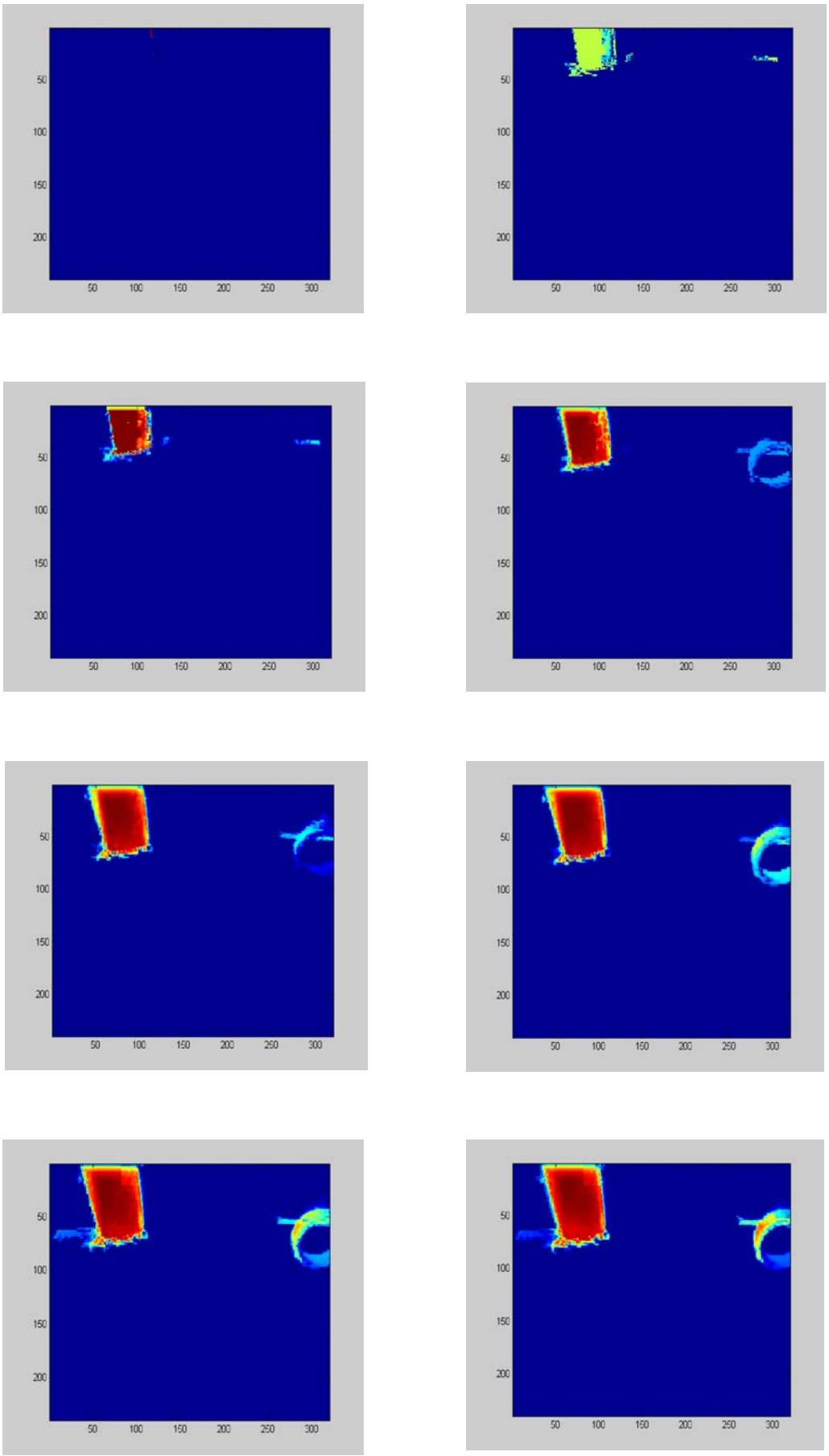


Fig. 4.5 Temporal Integration of Segmentations Results Increases Segmentation Accuracy if the Same Obstacles are Segmented in Multiple Frames

4.5 Computing the Distance to Obstacles

Once obstacles are detected it is desirable to determine the distances to those obstacles in order to generate an optimal trajectory which will be followed by the vehicle to navigate around the obstacles. In the introductory chapter it was assumed that all obstacles rest on the ground plane, that is, there are no overhanging obstacles[12]. For an obstacle-free ground plane distance from the robot is a monotonically increasing function of the vertical image coordinate (*v)*. This function is shown for a typical camera configuration in Figure 4.6(a). This function is called the depth map and for a camera pitch angle of 45° (zero roll and yaw angles) is independent of the horizontal image coordinate (*u*). The camera is fixed at 0.38 meters from the ground. The range in view of the camera at this configuration is approximately 0.65 meters, with the furthest point in view is about 1.1 meters from the camera. For all columns in the obstacle image of Figure 4.5 containing at least one non-zero element we may calculate the approximate range to the obstacle by looking up the range of the largest *v*-coordinate in the obstacle image, which by the previously stated assumption, corresponds to the point of contact with the ground.

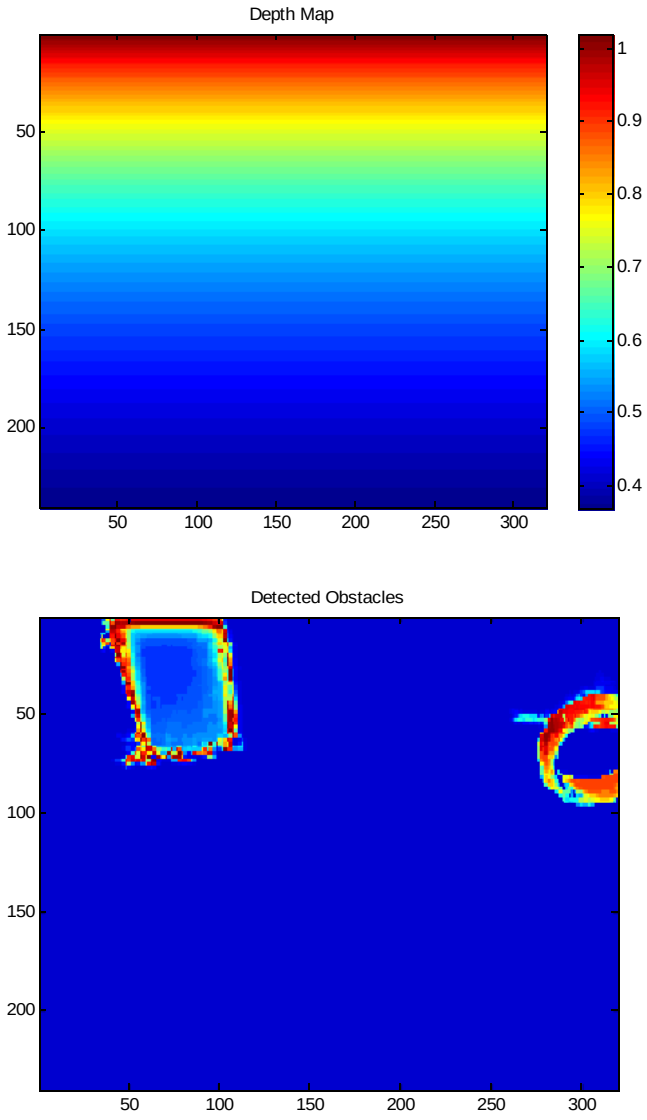


Fig. 4.6 Distance to camera as a function of image coordinates (top). Segmented obstacles (bottom).

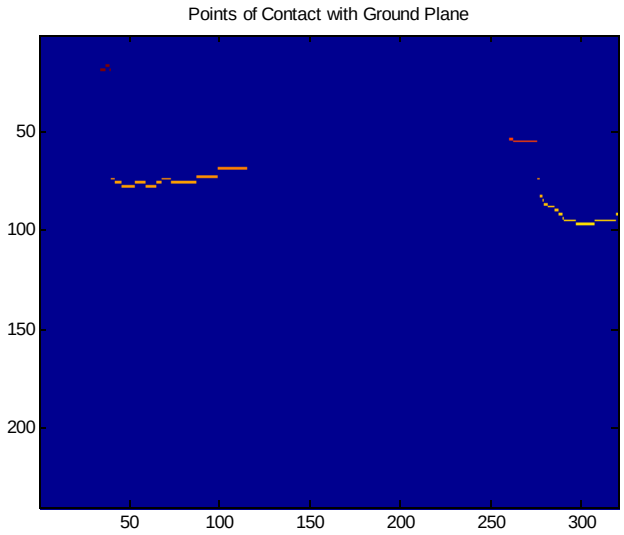


Fig. 4.7 Points of Contact between Obstacles and Ground Plane

For zero roll and yaw camera angles, the distance to an obstacle is a function of the vertical image coordinate. Therefore a graph may be constructed which represents the distance to the nearest detected obstacle as a function of the horizontal image coordinate. The distance graph represents the final output of the obstacle detection algorithm, this output may be used by a control algorithm to issue navigation commands to the vehicle.

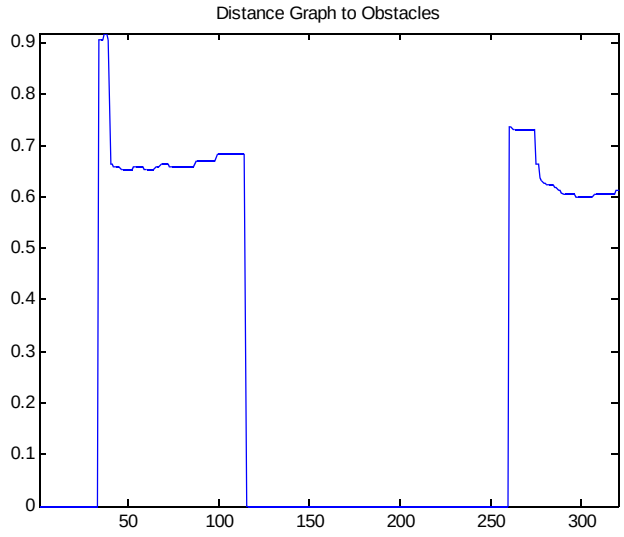


Fig. 4.8 Range to Obstacles as a Function of the Horizontal Image Coordinate.

CHAPTER 5

PRACTICAL IMPLIMENTATION

Issues regarding the practical implementation of the obstacle detection algorithm will be discussed in this chapter. The algorithm was implemented in MATLAB, and test data was post processed at an approximate rate of 0.5 frames per second. Test data consisted of a video sequence captured by a USB camera mounted on the moving robot as well as encoder and gyroscope inertial readings corresponding to the motion between adjacent frames in the video sequence. Appendix B presents a vision-based approach for measuring displacements. Although the approach discussed in this work was not implemented in closed loop on the robot, a simpler, appearance-based algorithm (see [12]) was implemented and operated in real time and in closed-loop on the robot. Some insights regarding the practical implementation of this appearance based algorithm carry over to the more elaborate algorithm discussed in this work.

This chapter will first describe the experimental setup followed by a discussion regarding camera calibration and synchronized data acquisition.

5.1 Overview of the System

The commercially available Pioneer 3-AT robot was used to conduct the experiments described in this work. An external computer interfaced to the robot via an RS-232 serial connection. Video data was captured by a basic serial (USB) color camera. The robot control program was written using ARIA (Advanced Robotics Interface Application[16])

and the video capture/processing program was written using the Open CV(Computer Vision[17]) library in the C++ language. Information between the control and vision applications was sent using the universal datagram protocol (TCP/IP-UDP). Off line simulation and processing was carried out in MATLAB. A diagram of the set-up including all the physical and virtual connections is shown in Figure 5.1, and a photograph of the actual setup is shown in Figure 5.2.

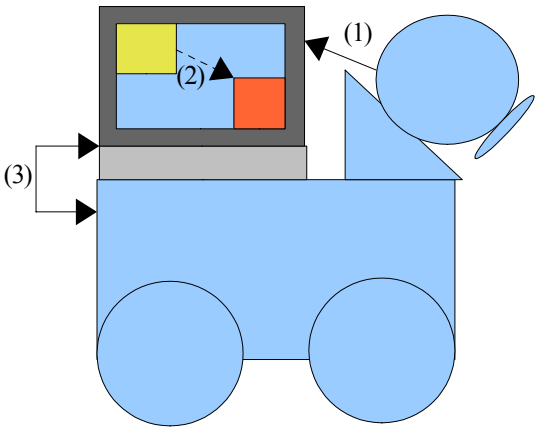


Fig. 5.1 Schematic of Virtual and Physical Connections. (1)USB Camera Connection (2)TCP/IP Virtual Link (3)RS-232 Serial Cable

Fig.5.2 Photograph of the Robot

5.2 Camera Calibration and Data Acquisition

Generation of the predicted image as described in Chapter 2 requires the knowledge of internal and external camera parameters. Internal parameters include focal length, and distortion parameters if optical distortion is significant [18]. External parameters refer to the camera placement and orientation, namely these are the lengths dZ_R and dY_R and the pitch(θ), roll(φ), and yaw(α) angles as described in Section 2.1. Furthermore, precise measurements of the robot motion between adjacent frames in the video sequence are required. External length parameters can be measured directly by measuring the distance from the center of rotation of the robot to the camera (dY_R) and the distance from the camera to the ground plane (dZ_R). Precise angle measurements are more difficult to obtain by direct measurement. The Caltech Camera Calibration Toolbox for MATLAB provides a method for measuring the orientation angles of the camera with respect to the ground plane using Rodrigues' rotation formula [18].

Displacement measurements between frames must be synchronized with the video data. Video data is recorded at variable frame rates due variable lighting conditions, the load on the central processor, among other variables beyond control. Therefore recording motion measurements at regular temporal intervals will yield motion readings which are misaligned with their corresponding frames in the video sequence. A method was devised for recording data such that the video data was synchronized with the displacement data. After a frame of video is received from the camera by the video processing program, a signal (flag) is sent over TCP/IP to the robot control program which, upon receiving this

flag demands the telemetry data from the robot and records it. In order to minimize the delay between the frame acquisition and displacement recording the decompression of the raw frame received from the camera is performed after the flag to record the telemetry data is sent. This Open CV code is shown below. Figure 5.3 shows two data acquisition methods; the blue arrows represent instances when position data is recorded at regular intervals, the red arrows represent instances when a flag is sent to record the position, and the black arrows represent when frames are received from the camera.

```
//Grab the Raw Frame
      cvGrabFrame(capture);
                    //Send Flag to Get Encoder Readings
                    j=sprintf(echoString_data, "%i %i",nframes, length);
                     sendto(sock, echoString_data, 100, 0, (struct sockaddr *)
                           &echoServAddr, sizeof(echoServAddr));
      //Now Decompress & Process the Raw Camera Frame
     frame = cvRetrieveFrame( capture );
            cvCopy(frame,frames[i]);
            cvFlip(frames[i],frames[i]);
    Sleep(500);
```

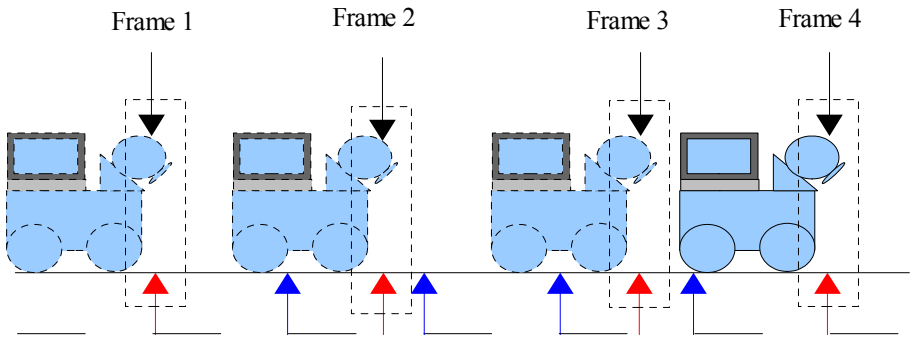


Fig. 5.3 Synchronous (red and black arrows) and Asynchronous(blue and black arrows) Data Acquisition Methods

CHAPTER 6

CONCLUSION AND FUTURE WORK

An attempt to design a general monocular obstacle segmentation algorithm for a GUV has been illustrated in this thesis. In the development of this algorithm basic assumptions about the operating environment of the ground vehicle were made, however initially no restrictions on the obstacle objects were placed. That is, we attempted to develop an algorithm which can be used to detect obstacles of any size, geometry or color.
For natural scenes, reliable obstacle segmentation was achieved. However, one may envisage scenarios where the algorithm will fail, for example due to seed pixel misalignment (Section 5.3).

This raises the fundamental question of whether a completely general obstacle segmentation algorithm is possible. Fundamentally, vast amounts of information are lost when the three dimensional world is projected onto the two dimensional image plane. Even when multiple views are available, depth information may not always be recoverable. To mitigate this loss of information constraints are often introduced. For example, the color uniformity constraint was introduced in order to infer connected regions representing segmented obstacles.

In the proposed approach obstacle segmentation was obtained by a procedure composed

of several discrete steps, namely: (i) computing the optical flow followed by (ii) Bayesian region growing. The optical flow computation is derived by minimizing an energy functional. One can envisage a new energy functional, which, when minimized, produces the desired segmentation. This would be equivalent to segmenting according to motion and intensity information simultaneously. This type of approach has been applied to motion segmentation by Cremers [21]. Furthermore, one can impose a temporal consistency penalty in the functional, in the spirit of the procedure described in Section 3.4, in order to obtain temporally consistent segmentations.

As stated in the introduction, this work was inspired by the fact that a human operator can detect obstacles with a high success rate in a monocular video sequence. However, in making the previous statement, the issue of human prior knowledge has not been addressed. Among the various cues used in human depth perception, scene interpretation is one of the most prevalent . That is, prior knowledge of the relative sizes of familiar objects along with an intuitive understanding of perspective allows humans to infer depth from two dimensional images . However, even the human capacity to infer depth may be compromised, as illustrated by many optical illusions [20].

Although generally ill-posed the problem of depth perception using a monocular camera is of great importance, especially in the the mobile robotics community. Obstacle detection and avoidance are basic robotic capabilities which, when solved, immediately allow for higher-level capabilities such as localization, mapping and path planning.

APPENDIX A

OPTICAL FLOW NORMALIZATION

In this appendix we will describe and justify a normalization applied to the flow field (between I_p and I_2) obtained by the optical flow computation. This normalization was applied to both images in Fig. 3.11 for valid comparison.

Consider the predicted flow field in Figure 3.3. The vectors at the bottom of the image have a larger magnitude than those at the top, therefore points at the bottom of the image will be transported over a larger distance than those at the top according to the planar prediction. We expect that because these points are transported over a greater distance their final intensity estimates will have greater error, in part due to interpolation error. This is true because the perspective projects inherently assigns less resolution to distant objects, see Figure 3.13. Therefore as the object moves closer in the view more details are observed which cannot be predicted by interpolation.

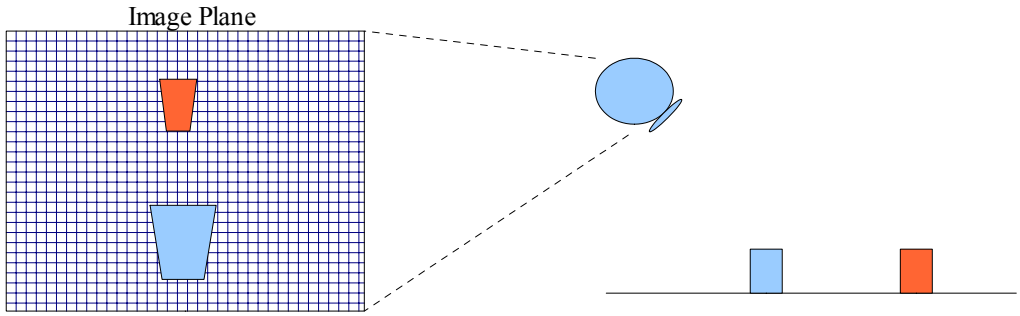


Fig. A.1 Distant objects are resolved more coarsely due to the perspective projection on a uniform image grid. Left picture shows the camera's view of the 3-D scene on the right.

Furthermore, because of the large displacement problem the optical flow calculation is expected to be more noisy for large displacement vectors. Other sources of noise include mis-calibration in the camera configuration, and displacement estimation errors, which also increase with the transport distance. Therefore in general we expect to have more noise in the predicted image where the motion vectors are larger. This hypothesis can be experimentally confirmed by computing the discrepancy from the planar prediction for a uniformly textured planar surface such as textured carpet. Ideally there should be no discrepancy between the predicted and actual images for such a planar surface, however the magnitude of the motion field produced by computing the optical flow between such images typically resembles that in Figure 3.15 (a), due to the sources of noise described above. Note that the discrepancy increase with the magnitude of the predicted transport vector (Fig. 3.3).

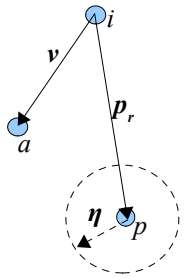


Fig. A.2 Illustration of the relationship between a point's initial location, its predicted location and its actual location after a camera displacement, in spatial-intensity space.

Let *i* be the initial spatial-intensity location of some point in the initial view. Let *a* be the actual location of the same point in the displaced camera view. Let $\boldsymbol{v}$ be the vector which transports *i* to *a*. Next, let *p* be the predicted location of *a,* which is the result of transporting *i* by $\boldsymbol{p}_r$, the vector corresponding to the planar prediction. Because the predicted transport of a point involves interpolation we expect an error (noise) in both the

intensity and location of the point. We combine all these sources of error into an additive noise vector term. The noise increases with the magnitude of $\boldsymbol{p}_r$, therefore we model the noise vector as the product of some base random vector $\boldsymbol{\eta}$ with the magnitude of $\boldsymbol{p}_r$.

The relationships depicted in Figure 3.14 are described in Equations 20-21.

$$a = i + \boldsymbol{v} \quad (20)$$

$$p = i + \boldsymbol{p}_r + \boldsymbol{\eta}\left\|\boldsymbol{p}_r\right\| \quad (21)$$

The computation described by Approach 2 attempts to solve for the error vector given by $\boldsymbol{O}=a-p$, however only the magnitude of this vector is important in the detection of non-planar objects.

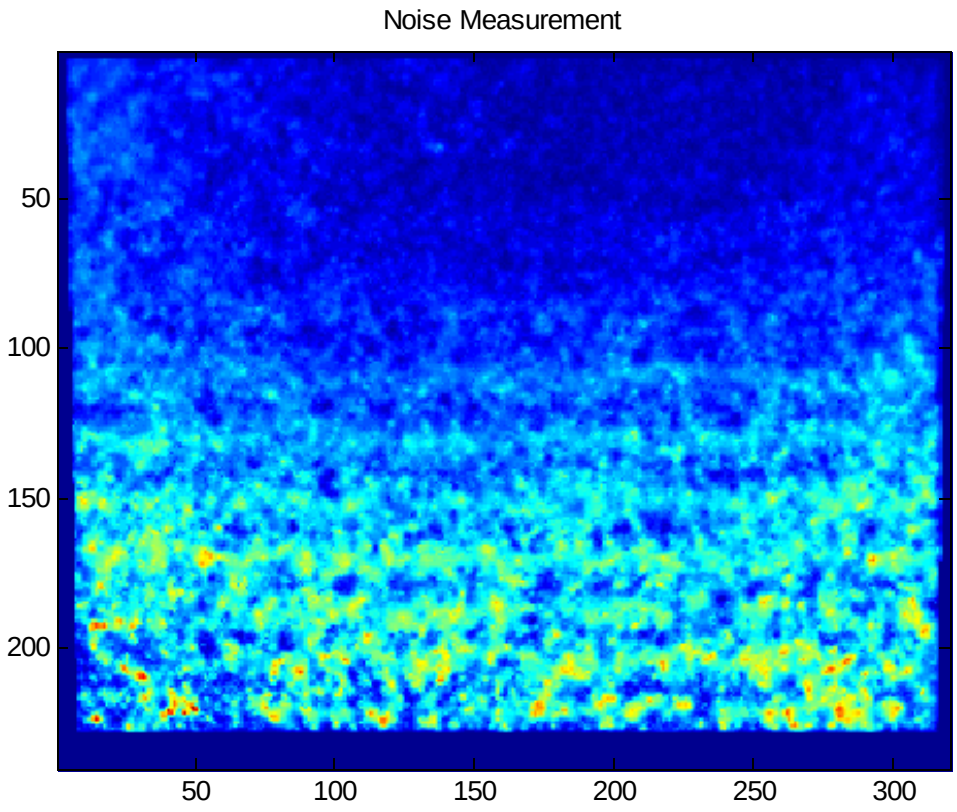
$$\left\|\boldsymbol{O}\right\| = \left\|a-p\right\| = \left\|(i+\boldsymbol{v})-(i+\boldsymbol{p}_r+\boldsymbol{\eta}\left\|\boldsymbol{p}_r\right\|)\right\| = \left\|\boldsymbol{v}-\boldsymbol{p}_r+\boldsymbol{\eta}\left\|\boldsymbol{p}_r\right\|\right\| \quad (22)$$

The vector $\boldsymbol{e}=\boldsymbol{v}-\boldsymbol{p}_r$ is the desired measurement and determines if the motion of a point conforms to the planar assumption, however it is corrupted by the additive noise term ($\boldsymbol{\eta}$). Suppose we introduce a normalization term $N=\left\|\boldsymbol{p}_r\right\|+1$ into the magnitude of the computed flow field. We add one in the expression for N to avoid singularities in the case where $\boldsymbol{p}_r$ is the null vector. The equation for normalized magnitude of $\boldsymbol{O}$, call it $\boldsymbol{O}_N$, is:

$$\left\|\boldsymbol{O}_N\right\| = \left\|\frac{(a-p)}{N}\right\| = \left\|\frac{(i+\boldsymbol{v})}{(\left\|\boldsymbol{p}_r\right\|+1)} - \frac{(i+\boldsymbol{p}_r+\boldsymbol{\eta}\left\|\boldsymbol{p}_r\right\|)}{(\left\|\boldsymbol{p}_r\right\|+1)}\right\| = \left\|\frac{(\boldsymbol{v}-\boldsymbol{p}_r)}{(\left\|\boldsymbol{p}_r\right\|+1)} + \frac{(\boldsymbol{\eta}\left\|\boldsymbol{p}_r\right\|)}{(\left\|\boldsymbol{p}_r\right\|+1)}\right\| \quad (23)$$

We do not know what the distribution of $\boldsymbol{\eta}$ is, or in what manner its mean and standard deviation increase with the magnitude of $\mathbf{p}_r$. We made the assumption that the magnitude of the random noise vector increases linearly with $\|\mathbf{p}_r\|$, therefore by dividing by a linear function of $\|\mathbf{p}_r\|$ we hope to make the noise term as uniform as possible throughout the motion field. Normalizing the motion field such that the noise is constant throughout

facilitates the identification of non-planar objects. Figure 3.11 (b) shows the normalized magnitude of the field and Figure A.3 (b) shows the unnormalized magnitude of the same field. Clearly the non-planar objects (obstacles) have greater contrast in the normalized image.



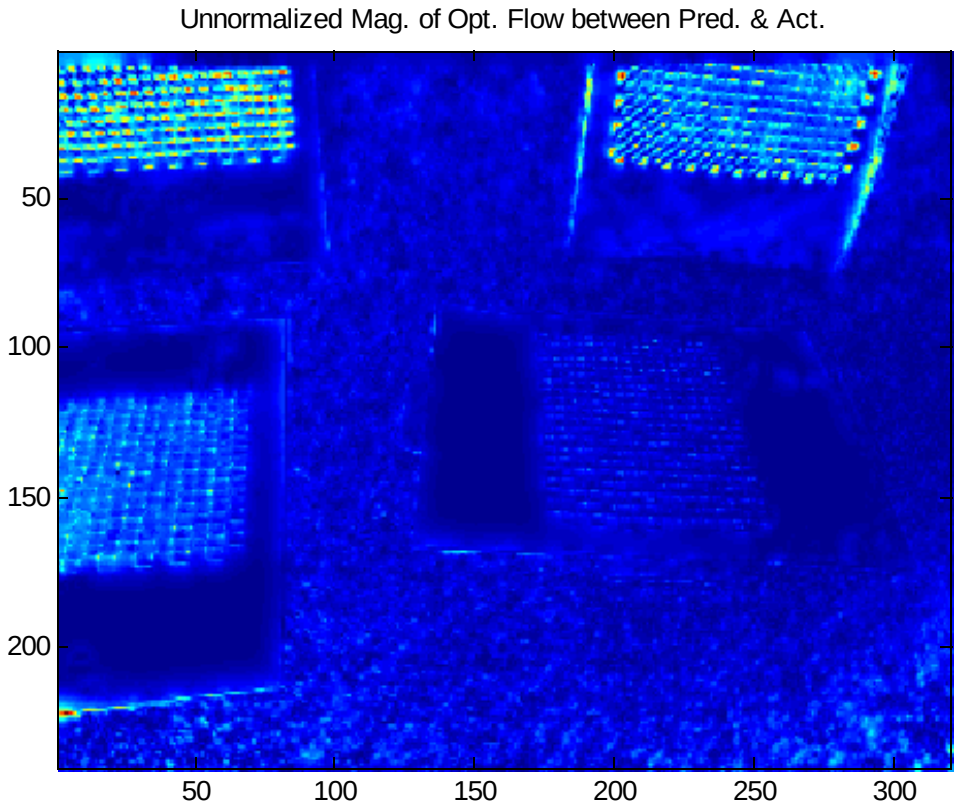


Fig. A.3 Top (a)Typical Noise Measurement. Bottom (b) Unnormalized Magnitude of Optical Flow

APPENDIX B

COMPUTING MOTION FROM VIDEO

It would be advantageous to have a mechanism to estimate the displacement between frames without relying on external sensors, such as encoders. For our purposes the displacement between frames must be measured very accurately in order to generate the correct predicted frame based on the planar assumption. This accuracy may be unattainable by the available sensors. In this appendix we present a general method for measuring the displacement between frames in a video captured by a moving camera. The procedure used to estimate camera motion is called bundle adjustment[19] and is widely used in the 3D computer vision community. The procedure simultaneously localizes one or more stationary reference points in space based on multiple projected views, while estimating the camera displacement between the viewpoints. In this appendix we will only simulate the first step of this procedure. Namely, given a set of projected views of some point in space, as well as a noisy estimates of the camera displacement between the views, we will attempt to estimate the 3D coordinates of that point. Once the coordinate of the this point is found, it may be used as a reference to estimate camera displacements.

Let (u,v) be the image coordinates of a point (X_0, Y_0, Z_0) viewed from a location described by the matrix g_c, relative to the previous camera configuration. The coordinates of the point relative to the displaced camera location are (X_1, Y_1, Z_1). The relationship between these points is described using homogeneous coordinates by the equations below. In this

appendix we assume a camera with unity focal length which leads to the normalized image coordinate ranges: $u \rightarrow [-1,+1]$ and $v \rightarrow [-1,+1]$. The origin is located at the center of the image.

$$\begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{bmatrix} = g_c \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} = \begin{bmatrix} r_{x1} & r_{x2} & r_{x3} & t_x \\ r_{y1} & r_{y2} & r_{y3} & t_y \\ r_{z1} & r_{z2} & r_{z3} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R_x} & t_x \\ \boldsymbol{R_y} & t_y \\ \boldsymbol{R_z} & t_z \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R_x} & t_x \\ \boldsymbol{R_y} & t_y \\ \boldsymbol{R_z} & t_z \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{P} \\ 1 \end{bmatrix} \quad \text{(B1)}$$

The coordinates of the projection of this point in the image plane of the current camera position are governed by the perspective projection equation described below.

$$\begin{bmatrix} u \\ v \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} \left(\dfrac{\boldsymbol{R_x P} + t_x}{\boldsymbol{R_z P} + t_z} \right) \\ \left(\dfrac{\boldsymbol{R_y P} + t_y}{\boldsymbol{R_z P} + t_z} \right) \\ 1 \\ 1 \end{bmatrix} \quad \text{(B2)}$$

We assume that the terms describing the camera motion ($\boldsymbol{R}$ and t terms) are only estimates of the actual camera motion. The variable we wish to solve for is $\boldsymbol{P}$; the 3D coordinates of the observed point. Equation B2 can be rearranged to yield the following system of linear equations for $\boldsymbol{P}$.

$$\begin{aligned} \left(u\boldsymbol{R_z} - \boldsymbol{R_x} \right) \boldsymbol{P} &= t_x - u t_z \\ \left(v\boldsymbol{R_z} - \boldsymbol{R_y} \right) \boldsymbol{P} &= t_y - v t_z \end{aligned} \quad \text{(B3)}$$

Observations of the point $\boldsymbol{P}$ from different camera configurations produces more equations. For every observation, we obtain a pair of Equations B3. The motion parameters ($\boldsymbol{R_n}$ and t_n) represent the estimated camera motion between the n and (n-1) observation. The motion parameters corresponding to the n^{th} camera motion is described relative to the camera configuration at the (n-1) observation. Assembling these into a

system of linear equations yields Equation B4.

$$\begin{bmatrix} u_1 \boldsymbol{R}_{z1} - \boldsymbol{R}_{x1} \\ v_1 \boldsymbol{R}_{z1} - \boldsymbol{R}_{y1} \\ u_2 \boldsymbol{R}_{z2} - \boldsymbol{R}_{x2} \\ v_2 \boldsymbol{R}_{z2} - \boldsymbol{R}_{y2} \\ \cdots \\ u_n \boldsymbol{R}_{zn} - \boldsymbol{R}_{xn} \\ v_n \boldsymbol{R}_{zn} - \boldsymbol{R}_{yn} \end{bmatrix}^T \boldsymbol{P} = \begin{bmatrix} t_{x1} - u_1 t_{z1} \\ t_{y1} - v_1 t_{z1} \\ t_{x2} - u_2 t_{z2} \\ t_{y2} - v_2 t_{z2} \\ \cdots \\ t_{xn} - u_n t_{zn} \\ t_{yn} - v_n t_{zn} \end{bmatrix} \quad \text{(B4)}$$

$$\boldsymbol{A}\,\boldsymbol{P} = \boldsymbol{b}$$

Because we assume that the point ***P*** is a stationary point in the environment, Equation B4 is an overdetermined system of equations for the point ***P***. The matrix **A** is full rank because we assume that each observation has a unique image coordinate. The least squares solution to an overdetermined system is given by the pseudo inverse operation, described by Equation B5.

$$\boldsymbol{P} = (\boldsymbol{A}^T \boldsymbol{A})^{-1} \boldsymbol{A}\,\boldsymbol{b} \quad \text{(B5)}$$

We have assumed that the projection of the point can be tracked perfectly from frame to frame. However, if the point is tracked by optical flow, or some other tracking procedure, we expect imperfect measurement of the point's image coordinate (*u,v*). Figure B1 (b) shows the apparent motion of a point (in blue) due to the forward motion of the camera. In red is the observed image coordinates of the point corrupted by a zero-mean noise process (simulating tracking error) used to estimate ***P***. In this simulation, estimates of the camera motion are generated by adding zero-mean Gaussian noise to the actual camera motion parameters. Figure B1 (a) shows the actual coordinate(blue) of the point relative to the camera (origin) and the estimated coordinate of the point in green. As the number of observations increases, the estimate of the world coordinate of the point converges.

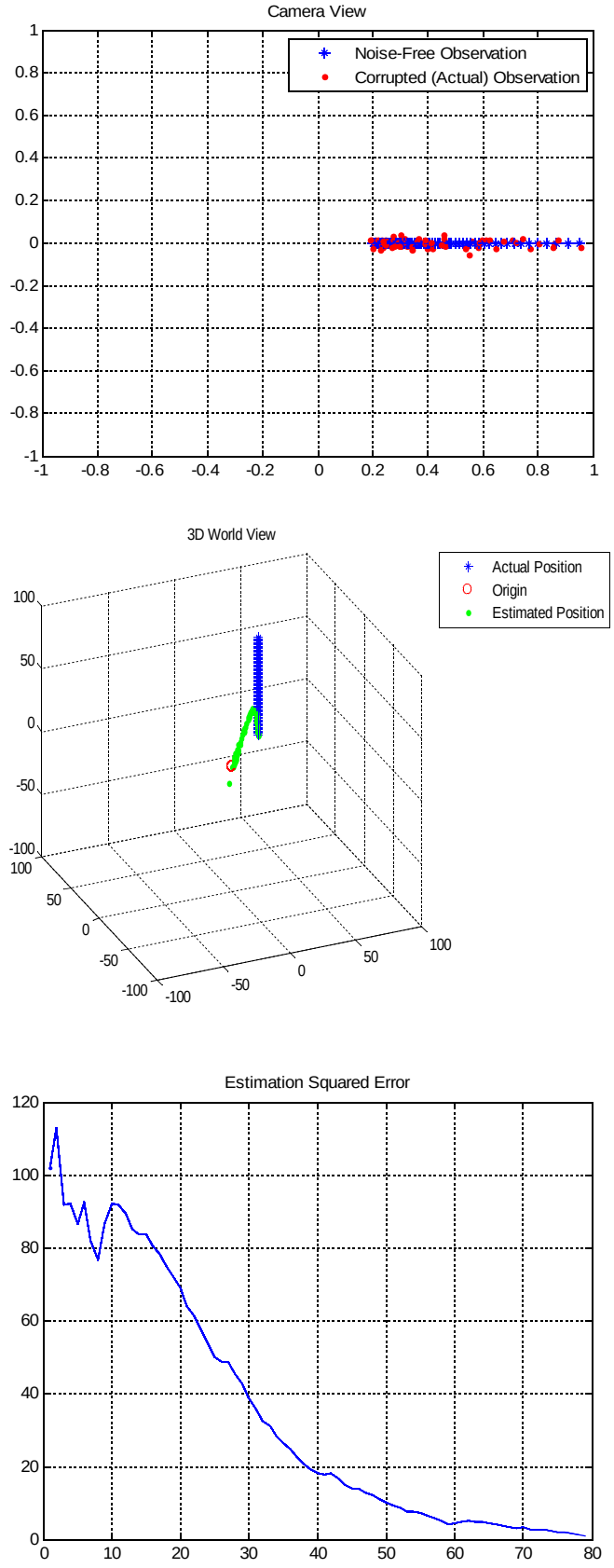


Fig.B.1 Estimating 3D Coordinates of a Point from Multiple Projected Views: Top 3D World View, Center: Camera View, Bottom: Squared Error as a Function of Number of the Number of Observations

REFERENCES

[1] Richard M. Murray, Zexiang Li S. Shankar Sastry “A mathematical introduction to robotic manipulation”. CRC Press New York

[2] Emanuele Trucco, Alessandro Verri “Introductory Techniques for 3-D Computer Vision”. Prentice Hall, 1998.

[3]Rfael C. Gonzales, Richard E. Woods. ”Digital Image Processing” 2nd Edition. Prentice Hall 2002

[4]Horn, B.K.P. , Schunck, B.G. “Determining optical flow” Artificial Intelligence, vol 17, pp 185-203, 1981

[5]PDE for Image Processing and Computer Vision (ECE6550) Spring 2008.
Dr. Anthony Yezzi Dept. of ECE Georgia Institute of Technology

[6]”Knowledge-Based Segmentation of SAR Data with Learned Priors” I*EEE* Transactions on Image Processing Vol.9 NO.2 February 2000

[7]”Variational Space-Time Motion Segmentation” Daniel Cremers and Stefano Soatto IEEE Intl. Conf. On Computer Vision (ICCV), Nice Oct. 2003 Vol 2 886-892

[8] “Region-Based Parametric Motion Segmentation Using Color Information” Yucel Altunbasak, P Erhan Eren, A. Murat Tekalp Graphical Models and Image Processing Vol 60 No.1 Januarry pp12-23 1998

[9]Martin Buehler (Editor), Karl Iagnemma (Editor), Sanjiv Singh (Editor) ”The 2005 DARPA Grand Challenge: The Great Robot Race” Springer 2007.

[10]*”Computer Vision: A Modern Approach”* David A. Forsyth, Jean Ponce Prentice Hall U.S. Edition 2002

[11]Ashuosh Saxena, Sung H. Chung, Andre Y. Ng. *”Learning Depth from Single Monocular Images”* NIPS 18, 2005

[12]”*Appearance-Based Obstacle Detection with Monocular Color Vision”* Iwan Ulrich and Illah Nourbakhsh AAAI Conference on Artificial Inteligence, Austin,TX July/August 2000

[13]*”A Resource-Efficient Approach to Obstacle Avoidance via Optical Flow”* M. Grunewald. Systems and Circuit Technology, Heinz Nixdorf Institute, University of Paderborn, Germany

[14] Jose Santos-Victor, Giulio Sandini. "*Visual-Based Obstacle Detection A purposive approach using normal flow"*

[15] Andreas Wedel, Thomas Schoenemann, Thomas Brox, Daniel Cremers. "*WarpCut-Fast Obstacle Segmentation in Monocular Video"* Computer Vision Group, University of Bonn, Germany.

[16]Mobile Robots ARIA Documentation. Accessed on: May 5th 2008 URL: http://www.activrobots.com/SOFTWARE/aria.html

[17]Open Computer Vision Library Accessed on: May 5th 2008 URL: http://opencvlibrary.sourceforge.net/CvAux

[18]Cal Tech Camera Calibration Toolbox Homepage . Accessed on May 5th 2008 http://www.vision.caltech.edu/bouguetj/calib_doc/

[19]W.E. Mantzel, H. Choi, and R.G. Baraniuk, "Distributed Camera Network Localization" Proceedings of the 38th Asilomar Conference on Signals, Systems and Computers, Nov. 2004

[20]Richard L Gregory "Knowledge in perception and illusion" Phil. Trans. R. Soc. Lond. B (1997), 1121-1128

[21]Daniel Cremers "A Variational Framework for Image Segmentation Combining Motion Estimation and Shape Regularization" Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition